Interactive Topology Optimization

Nobel-Jørgensen, Morten

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Interactive Topology Optimization

Morten Nobel-Jørgensen

Kongens Lyngby 2015

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

Interactivity is the continuous interaction between the user and the application to solve a task. Topology optimization is the optimization of structures in order to improve stiffness or other objectives. The goal of the thesis is to explore how topology optimization can be used in applications in an interactive and intuitive way. By creating such applications with an intuitive and simple user interface we allow non-engineers like designers and architects to easily experiment with boundary conditions, design domains and other optimization settings. This is in contrast to commercial topology optimization software where the users are assumed to be well-educated both in the finite element method and topology optimization.

This dissertation describes how various topology optimization methods have been used for creating cross-platform applications with high performance. The user interface design is based on theory of from human-computer interaction which is described in Chapter 2. Followed by a description of the foundations of topology optimization in Chapter 3. Our applications for topology optimization in 2D and 3D are described in Chapter 4 and a game which trains the human intuition of topology optimization is presented in Chapter 5. Topology optimization can also be used as an interactive modeling tool with local control which is presented in Chapter 6. Finally, Chapter 7 contains a summary of the findings and concludes the dissertation.

Most of the presented applications of the thesis are available at: `http://www.topopt.dtu.dk`.

Summary (Danish)

Interaktivitet er den løbende dialog mellem en bruger og et program for at løse en given opgave. Topologi optimering er optimeringen af strukturer for at øge deres stivhed eller andre egenskaber. Målet med denne afhandling er at udforske hvordan topologi optimering kan bruges i programmer på en interaktiv og intuitiv måde. Ved at skabe brugervenlige programmer til interaktiv topologi optimering vi når en større målgruppe, såsom designere og arkitekter, som kan eksperimentere med at skabe optimerede former ved at f.eks. at bruge understøtninger og laste. Dette står i skarp kontrast til kommercielt topologi optimerings-software, hvis brugere antages for at være eksperter på området.

Denne afhandling beskriver hvordan forskellige topologi optimerings metoder er blevet brugt til at lave programmer med høj performance til mange forskellige platforme. Design af brugerinterfacet er baseret på teori fra forskning i menneske-computer interaktion hvilket er beskrevet i kapitel 2. Topologi optimerings teori er beskrevet i kapitel 3. Vores topologi optimerings programmer i 2D og 3D er beskrevet i kapitel 4 og et undervisningspil som træner intuitionen omkring topologi optimering er præsenteret i kapitel 5. Topologi optimering kan også blive brugt som et interaktivt design redskab med lokal kontrol, hvilket er beskrevet i kapitel 6. En sammenfatning og konklusionen på afhandlingen findes i kapitel 7.

De fleste af de præsenterede programmer beskrevet i denne afhandling findes på følgende hjemmeside: `http://www.topopt.dtu.dk`

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark (DTU) in partial fulfilment of the requirements for acquiring the Ph.D. degree in engineering.

The thesis consists of a summary report and a collection of two published scientific papers and one paper currently under review. The work was carried out between 2012 and 2015.

Lyngby, 30-Juni-2015

Morten Nobel-Jørgensen

List of publications

Included papers

Paper A Aage, N., Nobel-Jørgensen, M., Andreasen, C. S., Sigmund, O. (2013). Interactive topology optimization on hand-held devices. *Structural and Multidisciplinary Optimization* 47.1: 1-6.

Paper B Nobel-Jørgensen, M., Aage, N. Christiansen, A. N., Igarashi, T., Bærentzen, J. A., Sigmund, O. (2014). 3D interactive topology optimization on hand-held devices. *Structural and Multidisciplinary Optimization* 51.6: 1385-1391

Paper C Nobel-Jørgensen, Morten, Malmgren-Hansen, D., Bærentzen, J. A., Sigmund, O., Aage, N. (2015). Improving topology optimization intuition through games. (Submitted)

Additional publications

- Aage, N., Nobel-Jørgensen, M., Andreasen, C. S., Sigmund, O. (2012) Interactive topology optimization. Presented at the 6th European Congress on Computational Methods in Applied Sciences and Engineering.
- Christiansen, A. N., Nobel-Jørgensen, M., Bærentzen, J. A., Aage, N., and Sigmund, O. (2013). Topology optimization using an explicit interface representation. Presented at the 10th World Congress on Structural and Multidisciplinary Optimization (WCSMO-10)

- Nobel-Jørgensen, M., Nielsen, J. B., Larsen, A. B. L., Olsen, M. D., Frisvad, J. R., Bærentzen, J. A. (2013) Pond of Illusion: Interacting through Mixed Reality Proceedings of SIGGRAPH Asia 2013 Posters
- Nobel-Jørgensen, M., Christiansen, A. N., Bærentzen, J. A., Aage, N., Sigmund, O. (2013) Improving Topology Optimization using Games Presented at the 10th World Congress on Structural and Multidisciplinary Optimization (WCSMO-10)
- Christiansen, A. N., Nobel-Jørgensen, M., Aage, N., Sigmund, O., and Bærentzen, J. A. (2014). Topology optimization using an explicit interface representation. *Structural and Multidisciplinary Optimization*, 49(3):387-399
- Christiansen, A. N., Bærentzen, J. A., Nobel-Jørgensen, M., Aage, N., and Sigmund, O. (2014). Combined shape and topology optimization of 3D structures. *Computers & Graphics*, 46(2015):25 - 35. Shape Modeling International 2014
- Malmgren-Hansen, D., Nobel-Jørgensen, M, (2015) Using 3D Models to Annotate SAR Images for Objective Segmentation Performance Measures (Submitted)

Acknowledgements

I would like to thank my supervisor Andreas Bærentzen for sharing his knowdledge and advices throughout the project. I would also like to express my gratitude to my co-supervisors Ole Sigmund and Niels Aage for their endless enthusiasm about topology optimization and for always having a solution or an explanation for the topology optimization challenges I have encountered. And to Niels Aage for giving a big hand with the implementation of the optimization kernels. I am also thankful for the collaboration and the discussions with my co-authors Asger Nyman Christiansen, David Malmgren-Hansen, Casper S. Andreasen and Takeo Igarashi.

I am grateful for five months I had at Tokyo University under supervision of Takeo Igarashi where meet a lot of brilliant people including but not limited to Lasse Laursen, Masaaki Miki, Hsiang-Ting Chen and Nobuyuki Umetani.

Thanks to Anders Clausen who implemented the server side of the TopOpt app used for 2D marching squares and 3D email-export. Anders Clausen and I also co-supervised the flexible void extension to the TopOpt App created by Nis Peter Lange. And to Asger Nyman Christiansen for creating OOCholmod with me.

I would also like to thank Timothy A. Davis and University of Florida for allowing us to use relevant parts of the SuiteSparse in non-LGPL compatible platforms (iOS/Android).

I am grateful for the financial support from the Villum foundation who made this project possible through the grant 'NextTop'. I would also like to express my

gratitude to the Otto Mønsted foundation for supporting my study financially on my external stay. Furthermore, I am grateful to the Technical University of Denmark for the financial support of the project and to Lenabot for keeping my caffeine level high.

I would like to thank my colleagues, the members of the Image Analysis and Computer Graphics group, for all the interesting discussions and fun times. A special thank goes to Asger Nyman Christiansen for his support and our close collaboration throughout the project.

Finally, I am deeply grateful for the support given by my friends and family. I would especially like to thank my girlfriend Christina.

List of Symbols

$\mathbf{f}$	Global load vector containing the nodal forces
$\mathbf{u}$	Global displacement vector
$\mathbf{K}$	Global stiffness matrix
$\mathbf{K_e}$	Element stiffness matrix
$\mathbf{E}$	Constitutive matrix
E_e	Elastic modulus (Young's modulus)
v	Poison's ratio
N_e	Element basis function
ε_x,ε_y	Normal strain in x-direction and y-direction
γ_{xy}	Shear strain
$\mathbf{B}$	Strain-displacement field
$\mathbf{x}$	Design variables: Element densities (for density based topology optimization). Displacement in normal direction (for DSC based topology optimization)
x_{min}	Minimum element density for density based topology optimization
x_{min}, x_{max}	Move limits for DSC based topology optimization
c	Compliance
v_{max}	Volume fraction
p	Penalization parameter used in SIMP
q	Penalization parameter used in RAMP
$\mathbb{A}$	Mapping from local stiffness matrix to global stiffness matrix
$\hat{H}_f^e$	Convolution operator
r_{min}	Radius of filter
m	Move limit used in optimality criterial method

B_e	Optimality condition used in the optimality criterial method
η	Numerical damping coefficient used in the optimality criterial method
λ	Lagrangian multiplier
P_{void}	Set of passive elements with forced void
P_{mat}	Set of passive elements with forced material
n	Normal direction
g_i	Constraints (such as volume constraints)

Note that the mathematical notation may deviate in the included papers.

Contents

Chapter 1

Introduction

Topology optimization is a structural optimization tool, which optimizes the material distribution within a design domain in order to maximize stiffness or other objectives. The research field of topology optimization started within structural and continuum mechanics but is now also used in other fields such as optics, acoustics and civil engineering. The method has since its introduction in the late 1980s been used in a number of different applications for optimizing structures such as airplanes, engines, buildings, antennas, optical fibers, and nanostructures. Recently, the method has received an increasing attention from architectural and industrial designers who wants to use the method as a shape design tool.

Topology optimization is a computational heavy task even when used on small problems. Previously solving small problems would take minutes or hours due to the limited amount of compute resources available. For this reason topology optimization was primarily used as batch processing, where the optimization problems was first defined and then send to the optimizer as a job that needed to be completed. The user would after job-completion evaluate the result and determine if the problem should be remodeled to obtain a different result.

Current commercial use of topology optimization is found in highly specialized software, usually as a plugin for computer aided design (CAD) systems or as an extension to Finite Element (FE) software. Commercial topology optimization

software is designed for expert users with a background in mechanical engineering who have received training in that particular software.

1.1 Motivation

The increase in computational power opens up new opportunities to solve computationally demanding tasks in an interactive manner, the same tasks that only a few years ago required long computation time. The increase in computation power has up until now roughly followed Moore's law [S.15] which states that the number of transistors per surface area doubles every second year.

The goal of this dissertation is to explore how this new capability of running topology optimization at interactive update rates can be used to create new and innovative applications. The focus is not on inventing new methods for topology optimizations, instead existing methods are carefully selected and used in new ways. Interactive topology optimization is a completely new way of thinking of and using topology optimization. The goal is to create simple and intuitive applications, which abstracts away some of the underlying complexity and make the applications more accessible for people outside the field, such as students, architects and designers. We hope to broaden peoples awareness by making topology optimization more accessible using intuitive applications running in web browsers, on smartphones or on tablets. By increasing the awareness of the research field we hope to increase the popularity of topology optimization such that it will be used to its full potential.

Additionally, the goal is to improve the way topology optimization is taught at universities. Often courses in the Finite Element Method (FEM) and topology optimization is taught using Matlab or using non-interactive FEM software. While this is a good approach for explaining the many details of the methods, we believe that truly interactive applications are much better for explaining the big picture as well as building students intuition about the methods. The interactive applications provides a constant flow of feedback based on the actions performed by the users. Using well-designed interactive applications as a teaching aid helps students come to a deeper understanding through "supporting conceptualization and contextualization", actively involving the student, and promoting internal reflections [CM01].

1.2 Thesis overview

Chapter 2 gives an overview of user interface (UI) design and user experience. The chapter focuses on the elements of Human-Computer Interaction (HCI) research relevant to this dissertation. The first section is a summary of design rules posed by several authors. The next chapter describes how knowledge learned from designing user interfaces can be captured and shared in catalogues of design patterns. The following section explains how to evaluate user interfaces using either UI experts or potential users. The chapter ends with a discussion of how the design rules are related and how the user interface design can be approached.

Chapter 3 gives an introduction to topology optimization mainly targeting people outside the field. The chapter starts with describing relevant concepts in solid mechanics followed by an overview of how the FEM can be used for structural analysis. Finally, topology optimization is introduced with a focus on the density approach and its concepts.

Chapter 4 describes the three interactive applications: TopOpt App (2D), Finger Finite Element Method (FFEM) and TopOpt 3D App. The chapter begins with introducing related work followed by a description of each of the applications. These descriptions include formulations of the problem, how the problems are solved interactively, and how the user interaction has been designed.

Next, chapter 5 describes how topology optimization has been gamified by creating the TopOpt Game. The chapter explains how gamification has been used in other research fields and how we have transformed topology optimization into an educational game. After discussing the game design and the implementation, the game data is analyzed in order to unveil how playing the game affects the players intuition of topology optimization.

Chapter 6 explains experimental research in how topology optimization can be used as an interactive modeling tool with local control. The chapter starts by introducing the Deformable Simplicial Complex (DSC) method followed by an overview of how topology can be optimized using DSC. After discussing the implementation detains the chapter summaries the findings and discusses what was learned.

Finally, the thesis is concluded in Chapter 7, which summarizes the thesis and its contributions.

In addition to the work on interactive topology optimization described in this dissertation, my other research publications are summarized in section D.

Chapter 2

User interface design

When engineers design computer programs to solve scientific problems the main focus is usually on performance or on the problem solving task. The user interface is often neglected and not considered important since the target audience is expert users who are willing to spend a significant amount of time on learning how to use the system.

Below are listed a number of different styles of user interface defined in a broad sense:

- **Source code.** Some projects are simply source code examples, where input are provided only by changing hardcoded values.
- **Software library.** Many project does not provide any user interface but instead allows interaction using a programming language. This has the benefit of easy integration with other systems. The downside is that it takes time to learn how to use the API and often requires documentation to be read.
- **Command line interface.** Allows easy integration with other programs using pipes and command line scripts. Command line programs are not suitable when many or complex parameters must be provided or when the

application is used in an interactive manner. Reading the documentation is usually needed in order to use such programs.

- **Configuration file.** Programs using configuration files usually allows much more sophisticated parameters than command line interface programs. Yet they are still not suitable for complex interactions. Again, the user needs to read the documentation to learn the valid configuration options.

- **Graphical user interface.** A well designed graphical user interface is often easy and intuitive to use. It has the benefit of providing good abstractions and gives visual hints which in best case allows the program to be used without consulting any documentation. However, interaction with other systems may be difficult.

Both source code as well as software libraries have the impediment of different compilers, operating systems, and third party library dependencies, which complicates running the program.

The consequence of not providing a suitable user interface is that the research is not used to its full potential. Providing only a poorly designed interface will scare students, newcomers and other non-experts, whom may end up using other research projects or students may find completely other research fields more appealing.

This chapter will discuss the foundation of how to create good user interfaces for non-experts for 2D and 3D problems within the mechanical engineering field based on research in Human-Computer Interaction (HCI).

2.1 Design rules

Creating good user interfaces is a complex task. Building the perfect interface requires knowledge of what kind of users you are targeting (sociology), the motivations, feelings and emotions of the user (psychology), the physical interaction (ergonomics), how the user solves problems using the application (cognitive science) - just to name a few of the areas of Human-Computer Interaction (HCI). While these are all important parts of good interface design, it is usually preferable to focus on design principles and user interface design patterns. By doing this we use the knowledge gained from the HCI research.

2.1.1 Alan Dix's principles to support usability

Dix et. al. [DFAB03] describe three general principles, which are the most abstract design rules than can be applied to the user interface design in order to ensure its usability:

- "**Learnability** - the ease with which new users can begin effective interaction and achieve maximal performance."
- "**Flexibility** - the multiplicity of ways in which the user and system exchange information."
- "**Robustness** - the level of support provided to the user in determining successful achievement and assessment of goals."

These abstract principles are often too vague to be useful on actual usability design, but excellent as categories for other more concrete design rules.

2.1.2 Golden rules of interface design

Shneiderman describes in his book [Shn92] eight golden rules for designing user interfaces. Awareness of these rules generally lead to better designs, even though the rules are general and cannot be used in every situation.

1. **Strive for consistency**. Consistency both in user interactions, terminology and visual layout.
2. **Design for all types of users**. Support all kinds of users. Novice users often need both hints, help, such as tooltip texts, and documentation, whereas the focus of experts users is on using the system efficient through system-responsiveness and keyboard shortcuts.
3. **Offer informative feedback**. For every action there should be a visible feedback.
4. **Design dialogs to yield closure so users know when a task is completed**. Design sequences of action to have a beginning, middle and an end.
5. **Offer error prevention and simple error handling**. Design the system such that the users cannot make serious errors. When this is not possible then guide the user to resolve the problems when they occur.

6. **Support undo of actions**. Every action should ideally be reversible.
7. **Make user feel in control of the system**. The users should be the initiators of actions rather than the responders to actions.
8. **Reduce short-term memory load**. Rule of thumb is that humans can remember "seven plus or minus two chucks" of information. Display relevant information when needed.

More broadly applicable design guidelines are found in Donald A. Norman's book [Nor02] describing how difficult tasks can be transformed into simpler tasks. He explains the importance of knowing when to use knowledge represented in the environment (here computer system) and when to use knowledge existing in the head. The book emphasizes the importance of the user building a good mental model of the system, and how the interaction should map well to this model. Good design should exploit the power of constraints to guide the user to complete the tasks and prevent him from making errors. When an error occur, the system should be able to recover from it and lead the user back on track. Finally, standardization can be used in situations when no natural mapping exists to enable users to reused knowledge about the interaction from similar systems.

Another comprehensive list of principles of interaction design can be found in Bruce Tognazzini's "First Principles of Interaction Design" [Tog14]. The list covers 19 design principles for traditional GUI applications, web applications, mobile devices, wearables and Internet-connected smart devices.

2.2 HCI patterns

Another way to work with user interface design is to use the experience gained by others when solving usability design problems. This experience is described in HCI patterns which are stored in pattern collections and pattern languages. HCI patterns are used similar to the way software design patterns are used in software engineering as described in for instance [GHJV94].

A HCI pattern is described using:

- **Pattern name** describing the essence of the problem, solution and consequence in one or two words. The name increases the design vocabulary and allows designers to easily refer to abstract concepts in a simple and unambiguous way.

- **Problem** explains the problem and the context in which it occurs.
- **Solution** describes the suggested solution including sub elements, responsibilities and reference patterns.
- **Consequences** discuss the trade-offs for applying the pattern. A typical tradeoff would be between learnability and expert usage. Consequences also discusses alternative patterns and distinctions from these.

Different pattern catalogues can have additional descriptive elements such as examples, links to usability research and advices of when to use the pattern.

It is important to emphasize that HCI patterns are always described with some level of abstraction such that they can easily be adapted to solve different design problems. Examples of HCI pattern catalogues and pattern languages are [Tid10], [Bor01] and [Gra03].

2.3 Usability evaluation methods

Evaluating the user interface design is an important step of software development. The goal of the evaluation is to identify usability problems and to ensure that the system enables the user in solving the intended tasks in an easy and intuitive manner. Finally, the evaluation is used to access the user's satisfaction with the system including the learning curve and the enjoyment of using the system.

Usability problems are often classified based on severity. There exists several different classifications such as Jeff Sauro's rating [Sau13] where each problem are rated as:

- **Minor**. Users hesitate or get slightly irritated.
- **Moderate**. Some users experience occasional task failure; causes delays and moderate irritation.
- **Critical**. Users fail to complete tasks and experience extreme irritation.

In addition to the usability problem rating, Sauro's approach also includes insights/suggestions/positives where users mention ideas or observations which could enhance the overall experience.

There exists two general types of usability evaluation methods (UEMs); Analytical methods where experts are analyzing the user interface and empirical methods where users are trying out the user interface to identify problems. Using expert evaluation is a relative cheap and fast method for evaluating the user interface using known heuristics and cognitive principles. On the other hand the problems found in empirical methods are actual usability problems. Empirical methods are also much better at quantifying the user's experience of the system is being used.

Over the years a huge number of UEMs have been developed. Four of the most popular are described below.

2.3.1 Heuristic evaluation

Heuristic evaluation is an informal method of usability analyses. Experts review the user interface using a simple set of heuristics in order to identify potential usability problems. Nielsen presents in [Nie95] a short list of ten heuristics for evaluation which strongly relates to the design rules from section 2.1. The heuristics are rules of thumb and may not always be applicable. Instead, the evaluation rely heavily on the experts own intuition and common sense. The heuristics can be summarized as:

1. **Visibility of system status.** Inform user of status using appropriate feedback within reasonable time.
2. **Match between the system and the real world.** Use intuitive abstractions that relates to familiar concepts.
3. **User control and freedom.** User should be able to change state easily and undo or redo actions.
4. **Consistency and standards.** Interface elements and interaction must be consistent. Use platform conventions.
5. **Error prevention.** Strive for error prevention by guiding user and eliminating error-prone conditions.
6. **Recognition rather than recall.** Objects, actions, and options should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use.** Support expert users by providing accelerators to speed up work.
8. **Aesthetic and minimalist design.** Keep the user interface simple.

9. **Help users recognize, diagnose, and recover from errors.** Error messages should be helpful and understandable for users.
10. **Help and documentation.** Systems should be designed to be used without documentation. But documentation should also be available and easy to understand.

A single evaluator usually only uncover under half of the usability problems. For this reason Nielsen and Molich suggests in [MB90] to aggregate usability problems from three to five people and to use other evaluation methods to identify remaining usability problems.

The main benefits of using heuristic evaluation are that the method is cheap, simple and intuitive to use. Besides, the method does not requires planning in advance and it can be used early in the development process. The downsides of using the method are that it does not suggest solutions for the identified problem and the problems identified may be biased by the mindsets of the evaluators.

2.3.2 Cognitive walkthrough

Cognitive walkthrough is a subjective and informal evaluation technique. Evaluators imagine themselves as users while performing known tasks in the system to be evaluated.

Cognitive walkthrough can performed using an incomplete prototype or in later stages of the development process. The tasks to be evaluated need to be documented including defining a list of actions to complete a given task. Finally, the experience and knowledge of target user need to be described.

While the evaluator is performing the tasks, he will for each action consider the following four things:

1. Is the effect of the action the same as the users goal at the time of executing the action?
2. Can the user see the action to be performed? Is it visible?
3. Does the user recognize the action to be performed? Is it clear and meaningful?
4. After executing the action, is the user able to see the result of the action? Is the feedback sufficient and clear?

Throughout the evaluation the good and bad parts about the user interface are documented - typically using a predefined evaluation form which includes a rating of the severity of the problem.

Usability evaluation with cognitive walkthrough was first described by John Rieman in [RFR95].

2.3.3 User based evaluation

In order to identify actual usability problems the evaluation needs to be performed using real users instead of usability experts. If real users are unavailable then a good alternative is to use test users matching the expected user profiles.

The classic evaluation method is to observe users perform one or more tasks in a controlled experiment. During the evaluation the observer documents any identified usability problem. After the evaluation the data is analyzed to reveal which of the usability problems are the most critical and need to be fixed.

The purpose of the controlled experiment is to answer a given hypothesis by measuring some attribute of participant behavior. Common attributes are time taken to perform the task and number of errors made. The experiment can formulated by using different configurations or alternatives (also known as independent variables), such as alternative interaction patterns or multiple user interfaces solving the same problem.

The evaluation can take place either in an UI laboratory or in the user's work environment. Laboratory studies have the benefit of having everything setup for usability tests, such as one-way mirrors, cameras, eye-trackers, microphones, etc. Their disadvantages are that the setup may not reflect the actual work environment of the user; there is no interruptions, no ambient noise and the situation may feel unnatural to the user. An evaluation performed in the user's own work environment may on the other hand be difficult to observe. A third option is to perform remote usability evaluation, where users and observers are partitioned in space and/or time [CHH98].

In general user based evaluation is more expensive than expert based inspections since it involves a larger number of people. Also analyzing the results and deriving the underlying problems takes a significant amount of time. The main benefit of using the method is that the found issues are actual usability problems.

2.3.4 Thinking aloud

In the thinking aloud method a test subject solves a series of realistic tasks using the system that is being evaluated. Throughout the evaluation the test subject is asked to "think aloud" by describing his intentions and understanding of the system while he is solving the tasks. The role of the evaluator (also known as the test monitor) is to passively observe while encouraging the test subject to share his thoughts. In the case of the test subject gets stuck solving a task, the evaluator may give hints or advices such that the task can be completed.

The advantages of the method are that it clearly points out usability problems and it uncovers parts of the users mental model of the system. The method is flexible and can be adapted to work even on paper prototypes (by letting the evaluator give the response of the system). Disadvantages are that the whole setup can feel a bit artificial, the test subject may feel that he is the one being tested and the passive role of the evaluator can also be challenging. The method also tend to focus on the learnability for novice users. Finally, after the usability evaluation an extensive analysis need to be performed to uncover the reasons behind the identified usability problems.

The method originates from Ericsson et. al.'s book [ES84] where it was developed for cognitive psychology to get valid data on people's thinking.

2.4 Platform constraints

A user interface design is also strongly influenced by the platform it runs on. This includes differences such as different screens in terms of both physical size and pixels per inch and well as interface devices like mouse, touchpad, keyboard, touch-screen, accelerometer, etc.

When designing for a wide range of devices such as PCs, smartphones and tablets, there may also be significant differences in terms of memory and compute power for both CPU and GPU.

The most obvious strategy for cross-platform development is to only use techniques that works well on all platforms. By doing this the users can easily switch between devices without worrying about what interaction technique to use. In some cases this strategy may be too restrictive; It may end up showing too much information on small screens or too little information on large screens. A good understanding of the headache that cross platform development can cause

is found in Jakob Nielsen's analysis of Windows 8 [Nie13], where he conclude that the user experience is "weak on tablets, terrible for PCs".

2.5 Comparing usability design rules and heuristics

The design rules from Dix et. al. [DFAB03], Shneiderman [Shn92], Norman [Nor02] and the heuristics from Nielsen [Nie95] can be combined as the following five rules:

1. **User supportive**. Applications should support users in doing their work efficiently.
    - Flexibility (Dix)
    - Design for all types of users (Shneiderman)
    - Make user feel in control of the system (Shneiderman)
    - User control and freedom. (Nielsen)
    - Flexibility and efficiency of use. (Nielsen)
2. **Consistent**. The user interaction and user interface should be designed in a minimalistic and consistent way using standards whenever available.
    - Learnability (Dix)
    - Strive for consistency (Shneiderman)
    - Standardization (Norman)
    - Consistency and standards. (Nielsen)
    - Aesthetic and minimalist design. (Nielsen)
3. **Informative**. Show relevant information when needed to communicate both the state of the system and the state of the tasks to the user
    - Learnability (Dix)
    - Offer informative feedback (Shneiderman)
    - Design dialogs to yield closure so users know when a task is completed (Shneiderman)
    - Reduce short-term memory load (Shneiderman)
    - Visibility of system status. (Nielsen)
    - Recognition rather than recall. (Nielsen)

- Help and documentation. (Nielsen)

4. **Design for errors**. Make it hard for the users to make errors. When an error occur, then the system should clearly show the error and offer ways to recover from the error.
   - Robustness (Dix)
   - Offer error prevention and simple error handling (Shneiderman)
   - Support undo of actions (Shneiderman)
   - Prevent errors by design (Norman)
   - Error prevention. (Nielsen)
   - Help users recognize, diagnose, and recover from errors. (Nielsen)
5. **Mental model**. Create a well-designed conceptual model of the system.
   - Learnability (Dix)
   - Knowledge in the head and in the world (Norman)
   - Match between the system and the real world. (Nielsen)

The three rules defined by Dix et. al. are very abstract and hence fit all the five categories. Shneiderman's eight golden rules are quite evenly spreads across the first four categories, but do not explicitly discuss the importance of a good mental model. Nielsen's ten heuristics are distributed on all five categories and are very similar to Shneiderman's eight rules. Finally, Norman's design guidelines are human-centric and covers all five categories even though only three bullet-points are listed above.

2.6 Discussion

Designing a good user interface is a complex task due to the many parameters involved. For this reason user interface design is usually designed in an iterative manner to find the most optimal design. These iterations happen both during the development of a single release of an application but also between different minor or major updates of an application. In many ways optimizing usability is similar to finding solutions for a non-convex optimization problem where we sometimes get stuck in a local optimum and other times learn valuable lessons that will help us reduce the search space.

For some types of applications the user interface design has converged and the frequently used design patterns are very obvious. Word processors are good

example this, where the user interface design is very similar across different applications and vendors.

When choosing between several user interface alternatives it is sometimes beneficial to select the most widely used and not necessarily the most well designed. Especially if learnability is a goal. If people have learned a bad habit there has to be a significant payoff for them to change their behavior without being annoyed. This is especially true when re-designing existing system with expert users as discussed in [JFY07].

Other types of applications have clearly not converged into a universal design. This is typical the case for applications with a small or narrow target group such or when dealing with very complex user interactions. One such example is applications manipulating 3D data. Even through there is some general trends, there are huge differences between applications and moving from one application to another application requires a significant effort and training for the end user.

Chapter 3

Topology optimization

This chapter will give an overall introduction to topology optimization by first introducing some solid mechanics foundations, then describing the finite element method and finally explaining topology optimization. The focus will be on optimizing for minimum compliance using a fixed amount of material, where the goal is to find a good solution with a high global stiffness.

3.1 Solid mechanics fundamentals

Solid mechanics is the study of the behavior of structures made of solid materials, especially their motion and deformation under the action of external forces.

We make the assumption that the materials we use are isotropic and linear-elastic. A supported structure of such material will deform when an external force is applied. The structure will return to its initial shape when the external force is removed. Many materials have this property as long as the magnitude of the force is under a certain limit (called the elastic limit).

When an external force is applied to a supported structure of an elastic material two things happen within the object:

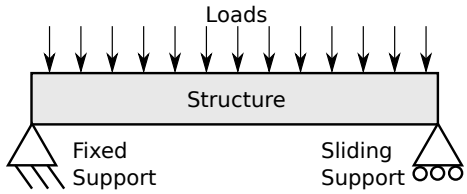


Figure 3.1: A solid mechanics problem showing a supported structure with loads applied.

- **Strain**: The relative deformation caused by the force.

- **Stress**: The restorative internal force, which makes sure that the object returns to its initial shape when the external force is removed.

We assume the material has a linear relationship between strain and stress. If twice the force is applied to a structure of such material, then twice the deformation will occur. A linear stress-strain relationship is also known as Hooke's law and can be stored in a constitutive matrix.

We also assume that the material is isotropic, such that the behavior of a structure is independent of the orientation of its material. Fortunately many materials, such as various metals, behaves both linear-elastic and isotropic.

Parts of the structure are fixed by rigid supports, which prevents the full structure from moving freely. From an optimization point of view the regions of supports can be seen as optional attachment points in order to prevent displacement. A support may be limited to only be active in certain directions. A classic example is when constructing a bridge, you would use fixed support in one end of the bridge and a sliding support (typically a roller) in the other. Using sliding supports the bridge allows small non-vertical movements under the support when forces are applied to the bridge which improves the stability of the bridge. An example of how such bridge can be modeled can be seen in figure 3.1. Supports are symbolized using triangles with legs or rollers.

External forces are often referred to as loads. Loads are usually applied to the boundary/surface of the structure. Point-loads are sometimes used when modeling a problem, however for real problems loads cover a larger area than an infinitesimal point. Loads are symbolized using arrows.

3.2 Structural analysis and the Finite Element Method

The core of topology optimization is structural analysis, where one of the characteristics, e.g. compliance, of a model is evaluated. To evaluate a structure of an arbitrary shape, the finite element method (FEM) can be used. The method is described in e.g. [C+07]. This method exploits that there exists analytical solutions for primitive shapes. The structure (or the full design domain containing the structure) is therefore discretized into elements of these primitive shapes with each element having a number of nodes attached. Using the analytical solution for each element as well as information of how each element relates to its neighbors, a solution for the discretized structure can be found. Due to details lost in the discretization of the structure the found solution is no longer exact.

The first use of the method was approximating a solution for the partial differential problem of structural analysis. The terminology used in FEM relates to concepts used within structural analysis, even though the method is now used as a general method for solving field problems where the goal is to find the spatial distribution of one or more dependent variables. The method is also known as finite element analysis especially when used for structural analysis.

The FEM is based on the static equilibrium equation, which states that the acceleration exerted by the external forces has to be countered by an equal and opposite directed force (Newton's first and second law). The equation can be formulated as follows

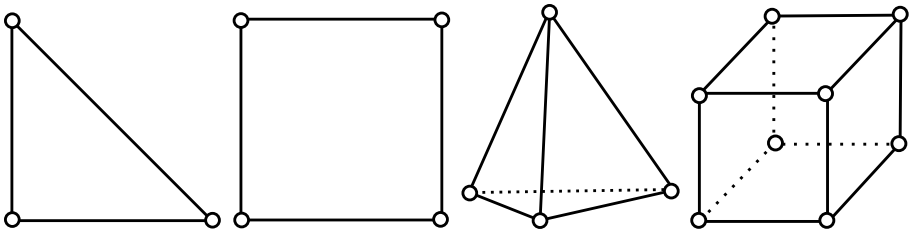
$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \tag{3.1}$$

, which states that the external forces $\boldsymbol{f} = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n]^T$ in a system must equal the displacements $\boldsymbol{u} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$ in a system with n nodes, where the stiffness relation between these nodes are expressed using the stiffness matrix $\boldsymbol{K}$ of size $n \times n$. Different problems can be solved this ways by using a different stiffness matrix derived for the particular problem.

We assume that the deformations are small enough that the equilibrium equations can be written using the original shape rather than using the deformed shape. We also only consider steady-state problems, also called static problems, which do not depend on time.

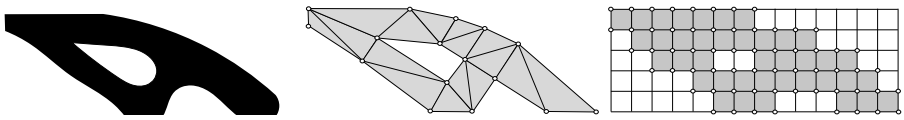
The FEM can be summarized to the following steps:

1. **Discretize**. This preprocessing step approximates the shape using simple



(a) Triangle **(b)** Quadrilateral **(c)** Tetrahedron **(d)** Hexahedron

Figure 3.2: Often used element types in FEM. The circles show the position of the nodes.



(a) Original structure **(b)** Unstructured FE mesh **(c)** Regular FE mesh

Figure 3.3: Discretization of a complex 2D shape into a finite element (FE) mesh using triangular and square elements.

element shapes with associated nodes.

2. **Build element stiffness matrix**. For each unique element shape we need to formulate the behavior of its nodes in matrix form based on the stiffness relationship between the nodes.

3. **Assemble global stiffness matrix**. Encode relations between all nodes in the global stiffness matrix.

4. **Define loads and supports**. Define a load vector. Modify global stiffness matrix to contain boundary conditions.

5. **Solve for displacements**. The linear system is solved in order to find the nodal displacements.

3.2.1 Discretization

When discretizing the problem commonly used elements are triangles and quadrilaterals for 2D models and tetrahedron and hexahedral for 3D models as listed in figure 3.2. Discretization approximates the original shape using a finite element mesh as shown in figure 3.3.

Each element contains a number of nodes, where each node contain one or more values about the system we are solving, such as displacements in the X, Y and Z direction. The nodal values are the unknowns in the linear system and they are often referred to as degrees of freedom or d.o.f. The nodes are shared between elements when located on the element boundary. The d.o.f. are used for determining the values within the element and when integrating over the element. For all elements in figure 3.2 the number of nodes equal the number of corners in the shape. Having only nodes in the corners allows creation of linear basis function used for interpolation of values inside the element. It is also possible to use more nodes for each elements, which allows creation of higher-order basis function, which gives a better approximation at the cost of more variables.

A regular FE mesh (figure 3.3-c) has the benefit that the basis functions are the same for every element as opposed to an unstructured mesh (figure 3.3-b) which requires custom basis functions for each individual element or a per-element mapping to fixed basis functions.

When using an unstructured FE mesh one has to pay special attention to the size and the shape of its elements. Since the mesh will be used to solve problems numerically, then numerical instabilities can occur if the size of an element is too small or the shape of an element is too flat. The finite element mesh needs to be of high quality, by having some constraints of minimum edge length as well as minimum and maximum angle between edges of an element. There also exists more formal ways to evaluate mesh quality, such as Radovitzky's tetrahedron-quality measure described in [RO00].

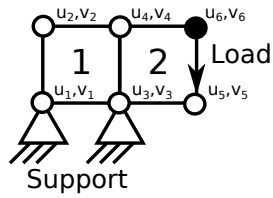


Figure 3.4: Example problem where elements and nodal d.o.f. are enumerated.

One important part of the discretization is to enumerate both the elements in the mesh as well as each d.o.f. for each node in the mesh. Figure 3.4 shows the enumeration of a 2D example where each node has two d.o.f. The enumeration is used when assembling the linear system.

3.2.2 Element behavior

For each element we need to define its local stiffness matrix, which describes the relationship between its nodal values.

The behavior is based on the stress-strain relationship, which can be formulated in a constitutive matrix. For the 2D case with no initial stresses the relationship is:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{3.2}$$

where $[\sigma_x\, \sigma_y\, \tau_{xy}]^T$ is the stresses, $[\varepsilon_x\, \varepsilon_y\, \gamma_{xy}]^T$ is the strains and $\mathbf{E}$ is relationship expressed in the constitutive matrix. The constitutive matrix $\mathbf{E}$ for an isotropic and plane stress condition is:

$$\mathbf{E} = \frac{E_e}{1-v^2} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & (1-v)/2 \end{bmatrix} \tag{3.3}$$

where v is Poison's ratio and E_e is the elastic modulus (Young's modulus), which are both properties of the given material.

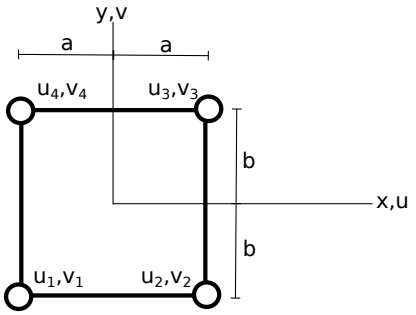


Figure 3.5: Quadrilateral element of size $2a \times 2b$ with eight d.o.f.

We define a basis function N_e for each node (also known as shape functions), which are used for interpolating d.o.f. values inside an element. As an example we use the 2D quadrilateral element with eight d.o.f. in figure 3.5. Notice how the d.o.f. also have a local enumeration. Based on this element we can define

the following bilinear basis functions:

$$\begin{aligned} N_1(x,y) &= \tfrac{(a-x)(b-y)}{4ab} \quad N_2(x,y) = \tfrac{(a+x)(b-y)}{4ab} \\ N_3(x,y) &= \tfrac{(a+x)(b+y)}{4ab} \quad N_4(x,y) = \tfrac{(a-x)(b+y)}{4ab} \end{aligned} \tag{3.4}$$

The complete element displacement field for this example is:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \sum_{i=1}^{4} N_i \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ \dots \\ v_4 \end{bmatrix} \tag{3.5}$$

While the displacement field describes the deformation within the element, the strain-displacement relation describes the strain field within the displacement field. We define normal strain (ε_x, ε_y) as change in length divided by normal length and shear strain (γ_{xy}) as change in right angle:

$$\varepsilon_x = \tfrac{\partial u}{\partial x} \quad \varepsilon_y = \tfrac{\partial v}{\partial y} \quad \gamma_{xy} = \tfrac{\partial u}{\partial y} + \tfrac{\partial v}{\partial x} \tag{3.6}$$

The same strain-displacement relation can also be formulated in matrix form:

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.7}$$

The strain-displacement field **B** for the element can now be found by combining the strain-displacement relation (equation 3.7), the element displacement field

(equation 3.5) and the basis functions (equation 3.4):

$$\mathbf{B} = \frac{1}{4ab} \begin{bmatrix} -(b-y) & 0 & -(a-x) \\ 0 & -(a-x) & -(b-y) \\ (b-y) & 0 & -(a+x) \\ 0 & -(a+x) & (b-y) \\ (b+y) & 0 & (a+x) \\ 0 & (a+x) & (b+y) \\ -(b+y) & 0 & (a-x) \\ 0 & (a-x) & -(b+y) \end{bmatrix}^T \tag{3.8}$$

Finally, we can compute the element stiffness matrix $\mathbf{K_e}$ by using the strain-displacement field $\mathbf{B}$ (equation 3.8) with the constitutive matrix $\mathbf{E}$ (equation 3.3) and the parameter t for element thickness (z-axis):

$$\underset{8\times 8}{\mathbf{K_e}} = \int_{-b}^{b}\int_{-a}^{a} \underset{8\times 3}{\mathbf{B}^T} \underset{3\times 3}{\mathbf{E}} \underset{3\times 8}{\mathbf{B}}\, t\, dx\, dy \tag{3.9}$$

The element stiffness matrix is usually determined analytically using a symbolic manipulation software, such as Maple.

3.2.3 Assembly

Using the element stiffness matrix we need to assemble the global stiffness matrix. The matrix contains the relations from each node to any neighbor nodes as showed in figure 3.6.

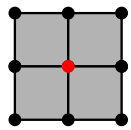


Figure 3.6: Single node with the adjacent elements and nodes.

The assembly is performed by adding each of the element stiffness matrices to the global stiffness matrix. Figure 3.7 shows this assembling including remapping values from the element d.o.f. index to the global d.o.f. index. This can also be

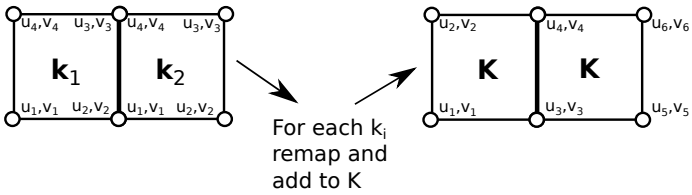


Figure 3.7: Assembling the global stiffness matrix based on element stiffness matrices.

expressed as:

$$\mathbf{K} = \sum_{e=1}^{n} \mathop{\mathbb{A}}_{e}(\mathbf{k_e}) \tag{3.10}$$

where $\mathop{\mathbb{A}}_{e}$ is an operator that maps from a local stiffness matrix (in this case 8×8) to a global stiffness matrix (in this case $n \times n$, with n being the global d.o.f.).

Figure 3.8 shows the conceptual global stiffness matrix $\mathbf{K}$ of the example problem. The light grey elements have coefficients from a single element stiffness matrix, whereas darker elements have coefficients from two element stiffness matrices. White elements contains the sparse regions with coefficients equal 0. For larger problems the matrix becomes significantly more sparse.

3.2.4 Defining loads and supports

Loads are always associated with nodes. A load has an associated force, containing both a direction and a magnitude, represented using a vector. The (global) load vector $\mathbf{f}$ is assembled by setting its values where nodal loads are defined. The remaining elements of the load vector are set to zero. For the example problem in figure 3.4 nodal values u_6 and v_6 are set to the load $[0, -1]^T$:

$$\mathbf{f} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{-1} \end{bmatrix}^T \tag{3.11}$$

For the boundary conditions defined by the supports, we need to enforce the displacements to be zero (if the node is supported in the given direction). This is done by modifying the global stiffness matrix by replacing rows and columns

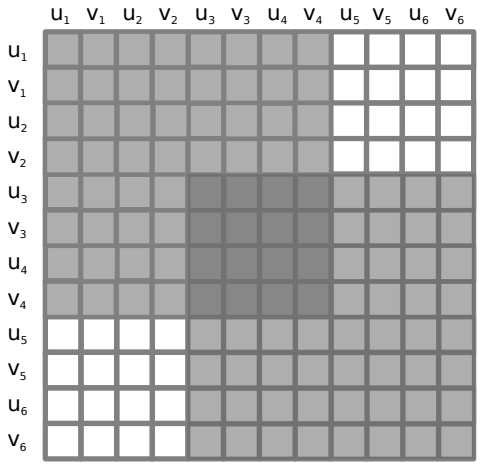


Figure 3.8: Assembly of the global stiffness matrix **K** of example problem from figure 3.4. The element colors represents usages, where white is unused (white)

of fixed nodes with the values of the identity matrix as showed in figure 3.9. As supported nodes cannot have loads applied, the displacements of such nodes are zero.

	u_1	v_1	u_2	v_2	u_3	v_3	u_4	v_4	u_5	v_5	u_6	v_6
u_1	1	0	0	0	0	0	0	0	0	0	0	0
v_1	0	1	0	0	0	0	0	0	0	0	0	0
u_2	0	0			0	0						
v_2	0	0			0	0						
u_3	0	0	0	0	1	0	0	0	0	0	0	0
v_3	0	0	0	0	0	1	0	0	0	0	0	0
u_4	0	0			0	0						
v_4	0	0			0	0						
u_5	0	0			0	0						
v_5	0	0			0	0						
u_6	0	0			0	0						
v_6	0	0			0	0						

Figure 3.9: Example of how the global stiffness matrix **K** is modified for the boundary conditions by enforcing the displacements of the supports to equal the value of the loads (which is 0 for these nodes).

3.2.5 Solving for displacements

The displacements are found by solving system of linear equations in equation 3.1 after load and support conditions have been defined.

$$\mathbf{u} = \mathbf{K}^{-1}\mathbf{f} \tag{3.12}$$

There are two main methods for solving these systems: direct sparse solvers and iterative solvers.

Direct sparse solvers are efficient on systems up to around 3 millions d.o.f. [Bat08]. High performance of the direct solvers is achieved by using graph theory to identify an optimal sequence to eliminate variables in the Gauss elimination when factorizing the stiffness matrix. One benefit of using direct solvers is that the factorization can be reused on multiple load cases.

Iterative solvers are usually used for larger systems. Iterative solvers have the benefit of being able to run in parallel on multiple cores. Iterative solvers are often implemented as multigrid solvers - sometimes combined with direct solvers.

3.2.6 Improving accuracy

There are several sources of errors when solving a problem using FEM. First of all, the mathematical problem which models the real-world phenomenon may be an oversimplification of reality. Solving the mathematical problem using FEM is also an approximation due to the discretization. The accuracy of FEM can be improved in two different ways:

H-method is the most obvious way to increasing accuracy by simply using more elements, which provides a better approximation of the original shape and captures more details.

P-method is increasing the accuracy by using more nodes per element, which will give higher-order basis functions for the interpolation and hence increase the accuracy.

Whether to improve the result by refining the discretization using the H-method or by using a more advanced model using the P-method depends on the nature of the problem being solved.

3.2.7 Performance considerations

Even through the FEM is a simplification of reality, all FEM problems, except textbook examples, are so large that they are not feasible to solve without using computers. For example a small FEM problem in a regular 3D grid of size 10^3 elements with each node having 3 d.o.f. (such as displacement in x, y and z direction). This small example will contain 3993 d.o.f. and solving it involves inverting or factorizing its stiffness matrix of size 3993×3993.

One important observation is that the stiffness matrix contains the direct relations between all nodes of the system. Since each node is only associated with its neighbor nodes (in adjacent elements), the stiffness matrix is a very sparse matrix. Using either a banded matrix or a sparse matrix representation will result in a significant performance gain and reduced memory footprint. A similar observation is that the relation between two nodes are symmetric. This means we can use a symmetric matrix representation, which will almost reduce storage size by half and improve computation time.

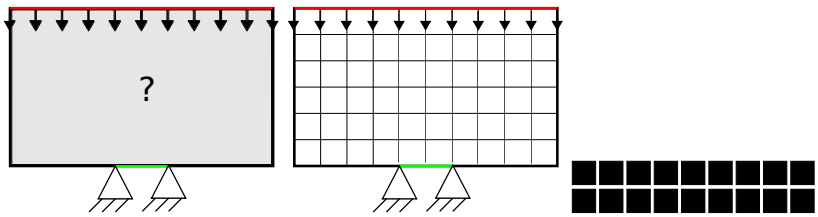
Another place to gain performance is when modeling the problem. Very often the problem and solution contains symmetry. E.g. if our small FEM problem described above contains symmetries in two dimensions, the elements are reduced to $10 \times 5 \times 5$ which means that the stiffness matrix is reduced to 1188×1188 elements, which reduces the number of coefficients in the (dense) stiffness matrix with $\approx 91\%$.

When FEM has to be evaluated multiple times, another choice is between using fixed grid or allow the structure of the grid to update in each iteration. By using a fixed grid, the stiffness matrix can be assembled once, factorized and reused in each iterations. Updating the element structure will on the other hand require the stiffness matrix to be rebuilt.

3.3 Topology optimization

Topology optimization can be formulated as an optimization problem using a FE formulation. Using a discretized design space of n elements, we are interested in distributing a fixed amount of material in order to optimize some objective function, such as compliance. Figure 3.10 shows how a topology optimization problem can be modeled and discretized.

The problem can be formulated mathematically as a non-linear mixed 0-1 prob-



(a) Model **(b)** Discretized model **(c)** Available material

Figure 3.10: 2D topology optimization using FEM discretization.

lem, where each element x_e can have a material density of either 1 (material) or 0 (void).

$$\begin{aligned} \min_{\mathbf{x}\in R^n} \quad &: \quad c(\mathbf{x}) = \mathbf{u}^T\mathbf{K}(\mathbf{x})\mathbf{u} && \text{(Compliance)} \\ \text{subject to} \quad &: \quad \mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f} && \text{(FEM equilibrium)} \\ &\quad\ \ \sum_{e=1}^{n} \frac{\mathbf{x}_e}{n} \leq v_{max} && \text{(Volume)} \\ &\quad\ \ \mathbf{x}_e \in \{0,1\} \quad e = 1,\ldots,n && \text{(Design variables)} \end{aligned} \tag{3.13}$$

where $c(\mathbf{x})$ is the compliance, $\mathbf{K}(\mathbf{x})$ is the global stiffness matrix, $\mathbf{u}$ is the displacement vector and v_{max} is the global volume fraction. Notice that the global stiffness matrix $\mathbf{K}$ depends on the element density $\mathbf{x}$.

Unfortunately we have no efficient way to solve the problem when it is formulated as a 0-1 problem. Using this formulation only very simple problems can be solved due to the large search-space containing $N!/((N-M)!M!)$ states, where N is the number of material elements and M is the total number of elements. For problems with 100 elements we can find the global optimum for some configurations, but not when N approaches 50 material elements as discussed in [SS03] and [SB11].

The workaround is to relax the problem to permit $\mathbf{x}$ to contain real numbers in the range from x_{min} (a small number larger than 0.0) to 1.0. This means that we allows each element to have a material density instead of the binary "material or void". x_{min} is used to avoid singularity. This formulation is called

the density approach:

$$\begin{aligned}
\min_{\mathbf{x}\in R^n} \quad &: \quad c(\mathbf{x}) = \mathbf{u}^T\mathbf{K}(\mathbf{x})\mathbf{u} && \text{(Compliance)} \\
\text{subject to} \quad &: \quad \mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f} && \text{(FEM equilibrium)} \\
&\quad\;\; \textstyle\sum_{e=1}^{n} \frac{\mathbf{x}_e}{n} \leq v_{max} && \text{(Volume)} \\
&\quad\;\; 0 < x_{min} \leq \mathbf{x}_e \leq 1 \quad e = 1, \ldots, n && \text{(Design variables)}
\end{aligned} \tag{3.14}$$

Topology optimization is usually solved using "the nested approach". That is, we first find the nodal displacements $\mathbf{u}$ using the finite element method. We then compute the sensitivity of the objective function for each element and change the material a step towards minimum compliance. And we repeat this until convergence (where changes of element densities are small). Listing 3.1 describes the main steps of the density approach in pseudo-code.

Listing 3.1: Topology optimization in pseudo-code.

```
1  // nelx, nely = number of elements in x and y direction
2  // volfract = volume fraction
3  // penal = penalization value
4  // rmin = filter radius
5  // maxChange = maximum change in X
6  Matrix topologyOptimization(nelx, nely, volfract, penal, rmin, maxChange) {
7      x = Matrix(nely, nelx)
8      x.fill(volfract)
9      do {
10         oldX = x
11         u = finiteElementAnalysis(x, penal, nelx, nely)
12         compliance = objectiveFunction(x, u)
13         dc = sensitivityAnalysis(x, u)
14         sdc = smoothingSensitilities(nelx, nely, rmin, x, dc)
15         x = optimalityCriteriaBasedOptimization(nelx, nely, x, volfrac, sdc)
16         change = maxDiff(oldX, x)
17     } while (change > maxChange)
18     return x
19 }
```

3.3.1 Penalization

One problem with the density approach stated in equation 3.14 is that the solution at convergence often contains a lot of gray-scale elements. From a manufacturing perspective gray elements are usually also not desirable. Most manufacturing processes, such as molding, require or prefer using a single material with uniform density. One noteworthy exception is 3D printers, which is capable of creating infills with different volume fractions.

To reduce the occurrence of grayscale elements, the density values are penalized such that they tend to become either brighter or darker.

One of the most popular penalization schemes for topology optimization is Simplified Isotropic Material with Penalization (SIMP) [Ben89], [ZR91], [Mle92] - also known as the power-law approach:

$$E(x_e) = x_e^p \tag{3.15}$$

where x_e is the element density and $p > 1$ is the penalization factor ($p = 3$ has been found to work well [BS99]).

Using SIMP the compliance part of equation 3.14 is changed to:

$$c(\mathbf{x}) = \sum_{e=1}^{N} (x_e)^p \mathbf{u}_e^T \mathbf{k}_e \mathbf{u}_e \tag{3.16}$$

where $\mathbf{k}_e$ is the element stiffness matrix with unit Young's modulus and $\mathbf{u}_e$ is the element displacement vector.

An alternative to SIMP is Rational Approximation of Material Properties (RAMP) [SS01]:

$$E(x_e) = \frac{x_e}{1 + q(1 - x)} \tag{3.17}$$

where $q > 0$ is the penalization variable. RAMP was introduced to improve non-concavity of SIMP interpolation, however it does not seem to play a strong role for practical problems [SM13]. Examples of different penalization variables in SIMP and RAMP can be found in figure 3.11.

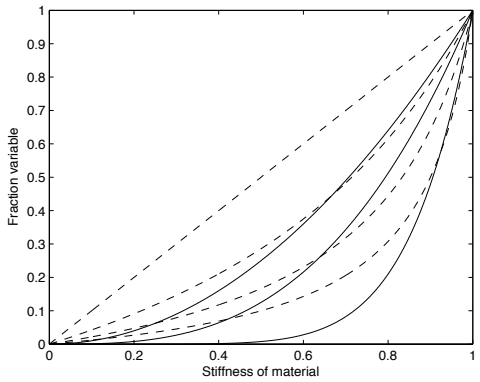


Figure 3.11: Dotted lines: RAMP with penalization= [0, 1, 4, 8].
Solid lines: SIMP with penalization= [2, 3, 7].

A variation of the SIMP worth mentioning is the modified SIMP as described in [Sig07]. This approach allows element densities $\mathbf{x}$ to be zero and instead adds a minimum density in the penalization:

$$E(x_e) = E_{min} + x_e^p(E_0 - E_{min}) \tag{3.18}$$

The modified SIMP has a number of benefits, such as simple implementation of additional filters.

For the remaining part of the chapter SIMP is used as penalization scheme.

3.3.2 Computing displacements and evaluating compliance

To compute the displacement vector $\mathbf{u}$ we need to assemble the global stiffness matrix $\mathbf{K}$ using the penalized element densities to solve the FEM equilibrium equation 3.1:

$$\mathbf{K} = \sum_{e=1}^{n} \underset{e}{\mathbb{A}}(x_e^p \mathbf{K_e}) \tag{3.19}$$

where $\underset{e}{\mathbb{A}}$ is the FE assembly operator.

After the global stiffness matrix $\mathbf{K}$ has been assembled and modified using boundary constraints, the displacement vector can easily be evaluated:

$$\mathbf{u} = \mathbf{K}^{-1}\mathbf{f} \tag{3.20}$$

Using the displacement vector the compliance of the current material distribution can be computed using equation 3.16.

3.3.3 Sensitivity analysis

The sensitivity of the objective function with respect to element densities x_e can be derived using the adjoint method with the result:

$$\frac{\partial c}{\partial \mathbf{x}_e} = -p(\mathbf{x}_e)^{p-1}\mathbf{u}_e^T\mathbf{k}_e\mathbf{u}_e \tag{3.21}$$

The full derivation can be found in [BS03]

The sensitivity of the material volume V with respect to element densities x_e when assuming unit area of elements is given by:

$$\frac{\partial V}{\partial \mathbf{x}_e} = 1 \tag{3.22}$$

3.3.4 Regularization

To ensure existence of solutions and avoid checkerboard patterns a simple smoothing filter is applied to the sensitivities as described in [Sig97]:

$$\widehat{\frac{\partial c}{\partial \mathbf{x}_e}} = \frac{1}{\mathbf{x}_e \sum_{f=1}^{n} \hat{H}_f} \sum_{f=1}^{n} \hat{H}_f \mathbf{x}_f \frac{\partial c}{\partial \mathbf{x}_f} \tag{3.23}$$

where $\hat{H}_f$ is the convolution operator (weight factor) which can be written as:

$$\hat{H}_f = max(0, r_{min} - \mathrm{dist}(e, f)) \tag{3.24}$$

where r_{min} is the filter radius and $\text{dist}(e, f)$ is the distance between element e and element f. The filter weight decays linearly from the element e to a radius of r_{min}.

As an alternative, or supplement, to smoothing the sensitivities, a smoothing filtering can be run on element densities $\mathbf{x}$ using a filter such as:

$$\tilde{x}_e = \frac{1}{\sum_{f=1}^{n} \hat{H}_f} \sum_{f=1}^{n} \hat{H}_f \mathbf{x}_f \tag{3.25}$$

3.3.5 Optimization

The algorithm now updates the element densities x_e using the Optimality Criteria (OC) method. This method is based on the following heuristic for how material is redistributed based on the current element density and the element sensitivity:

$$x_e^{new} = \begin{cases} max(x_{min}, x_e - m) \\ \quad \text{if} \quad x_e B_e^{\eta} \leq max(x_{min}, x_e - m) \\ \\ x_e B_e^{\eta} \\ \quad \text{if} \quad max(x_{min}, x_e - m) < x_e B_e^{\eta} < min(1, x_e - m) \\ \\ min(1, x_e - m) \\ \quad \text{if} \quad min(1, x_e - m) \leq x_e B_e^{\eta} \end{cases} \tag{3.26}$$

where m is the positive move limit, B_e is the optimality condition and η is the numerical damping coefficient (such as $\frac{1}{2}$).

The optimality condition B_e is defined as:

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}} \tag{3.27}$$

, here λ is the Lagrangian multiplier that can be found by a bisection algorithm which ensures that the global volume constraint is satisfied.

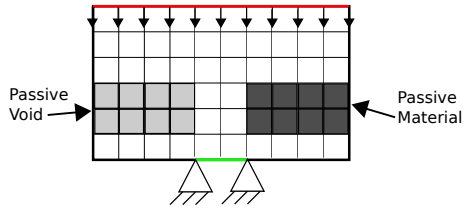


Figure 3.12: Passive elements in topology optimization.

As an alternative to the OC method there exists a number of other optimizers such as the Sequential Linear Programming method (SLP) described in [YC94] or the Method of Moving Asymptotes (MMA) described in [Sva87].

3.3.6 Passive elements

Topology optimization using the density approach is often computed on top of regular rectangular grids. This simplifies the implementation of the FEM mesh and hence also has a positive effect on the final runtime complexity. To allow non-rectangular design domains, certain elements can be flagged to be passive (either enforcing material or void) as shown in figure 3.12. The optimization should then skip updating element densities on any element flagged as passive. The density update heuristic can be modified to the following:

$$x_e^{new} = \begin{cases} x_{min} \\ \quad \text{if} \quad e \in P_{void} \\ \\ 1 \\ \quad \text{if} \quad e \in P_{mat} \\ \\ max(x_{min}, x_e - m) \\ \quad \text{if} \quad x_e B_e^\eta \leq max(x_{min}, x_e - m) \\ \\ x_e B_e^\eta \\ \quad \text{if} \quad max(x_{min}, x_e - m) < x_e B_e^\eta < min(1, x_e - m) \\ \\ min(1, x_e - m) \\ \quad \text{if} \quad min(1, x_e - m) \leq x_e B_e^\eta \end{cases} \tag{3.28}$$

where P_{void} and P_{mat} is the set of passive elements with forced void or material.

3.3.7 Iterations and converging

As outlined in listing 3.1 the optimization is run until convergence where no significant change of the structure occurs. The maximum change of element density is used as stopping criteria which will end the optimization when the change is under a given threshold.

3.3.8 Examples

The Messerschmitt–Bölkow–Blohm (MBB) beam is one of the standard problems used for topology optimization. The problem can be described using a design domain with the width-to-height aspect of 3:1. The problem has a downward load, in the upper left corner, vertical rolling support at the left side and a fixed support in the lower right corner, as shown in figure 3.13.

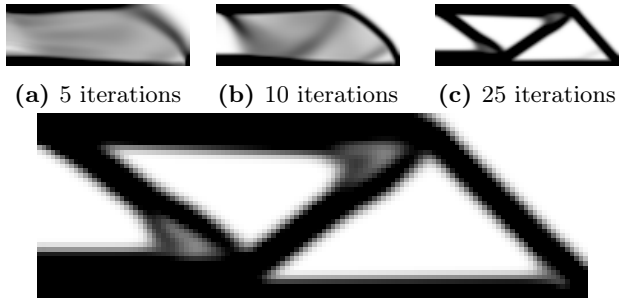
Figure 3.13: Example problem.

The volume fraction v_{max} is set to 0.5, which allows up to half of the domain to be material, and the filter radius r_{min} is set to 3.0. A SIMP interpolation model is used with a penalization factor p of 3.0.

Figure 3.14 shows the example problems running for 100 iterations before the maximum change in a single element density is below 0.01 after the 100th iteration. After the first 10 iterations the final topology begins to appear and the distinction between the 25th and the 100th iterations is hardly visible.

The result can clearly be interpreted as a truss structure with triangular holes.

As can be seen from figure 3.14 the result still contains some amount of gray elements, even through they are penalized using SIMP. An often used approach to extract the surface from element densities is to use marching squares for 2D or marching cubes for 3D [LC87]. Note that this is an interpretation of the result and the structural performance of the extracted shape may be different from



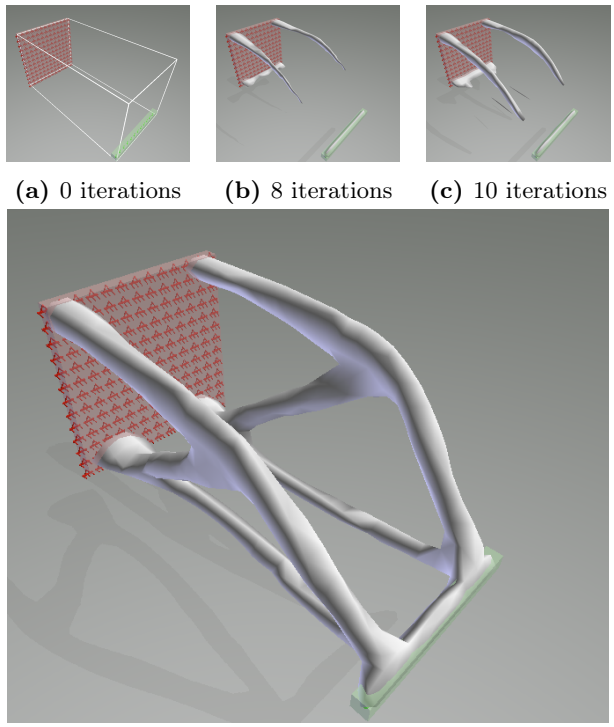
(a) 5 iterations **(b)** 10 iterations **(c)** 25 iterations

(d) 100 iterations

Figure 3.14: The result of the example problem during the first 100 iterations.

the performance computed using the element densities. Also the constraints are no longer guaranteed to be satisfied.

A similar problem can be posed in 3D as seen in figure 3.15. Here the result is visualized using marching cubes since visualizing densities of 3D elements is hard. The downward loads are located at the green bar in lower right corner and the supports are located at the the red plane. Since the optimization starts with the density of all elements set to the material fraction, the marching cubes surface starts without finding any shape. As the element density increases after a few iterations, the shape starts to appear. Even though the appearance of the shape in figure 3.15 looks smooth, the underlying FE mesh is only $24 \times 12 \times 12$ elements.



(a) 0 iterations **(b)** 8 iterations **(c)** 10 iterations

(d) 17 iterations

Figure 3.15: The result of the example problem during the first 17 iterations.

Chapter 4

Interactive topology optimization in 2D and 3D

This chapter shows new interactive ways of using topology optimization in applications. The main focus is interactivity and usability. Interactivity requires that the application is responsive and the update rate of the optimization is high. This is achieved by using highly optimized code, by using only the fastest methods and by limiting how large problems are solved. The update rate is kept under 10 seconds, which is "about the limit for keeping the user's attention" as discussed in [Nie93]. Another challenge is how to design a simple, yet flexible, user interface for interactive topology optimization. The user interface should additionally work equally well when running on PCs and on handheld devices such as smartphones and tables.

This chapter first gives an overview of work related to these topics. It then describes how interactive topology optimization has been accomplished for 2D problems in the TopOpt 2D app - also the topic of Paper A. The following section explains how this app has been used as a starting point for the Finger Finite Element Method app. The next section describes how the computationally harder problems are solved in the TopOpt 3D app as well as how its user interface has been designed. The TopOpt 3D is also the topic of Paper B. Finally, the last section evaluates the success of the applications and discusses choice of software platform and software architecture used.

4.1 Related work

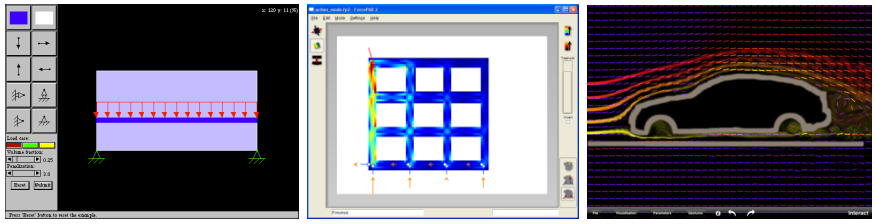
The precursor of the interactive TopOpt apps is the web-based topology optimization program created and hosted by the TopOpt group [TS01]. This application uses a Java Applet frontend for modeling the problems and visualizing the results as shown in figure 4.1a. The actual computations are performed on a server backend. The interaction pattern is batch processing where the user submit the optimization job to the server and wait 5-10 seconds until the results are received and showed as short animation. The application was very innovative when it was launched in 1999 by allowing users to perform topology optimization directly in the browser. The application has been a huge success with over 13,000 unique users during the first 13 years. However, compared to todays standards its user interface and the problem sizes seems a bit outdated.

ForcePad is a similar tool for modeling, analyzing and solving 2D structural problems. Figure 4.1b shows Von Mises stress visualized using ForcePad. The tool performs all computations locally on the users PC and is able to solve detailed problems. The application can be used for computing displacements, Von Mises stress as well as performing topology optimization. Even through the computations are performed locally in an efficient manner, the application still has a distinct modeling mode and an action mode where the computations are performed and visualized. ForcePad is created by the division of Structural Mechanics in Lund University and is released under an open source GPL license. The application and its source code is freely available at `http://forcepad.sourceforge.net/`.

An example of a more interactive physics simulations is found in apps performing aerodynamic analysis using flow simulations. One such app is WindTunnel available at `http://www.algorizk.com`. This app allows you to draw a custom shape during the wind simulation and instantly see a visualization of the effect. The application can be used for detecting high and low pressure fields, identifying turbulence and computing wind speed. An example usage is shown in figure 4.1c.

4.2 TopOpt 2D

The TopOpt app allows modeling a topology optimization problem while the program continuously optimizes the shape based on the current configuration. This interactive modeling as well as the frequent updates of the shape gives a feel of the shape becomes alive during the modeling. Besides, due to the short



(a) Legacy TopOpt Applet **(b)** ForcePad **(c)** Windtunnel

Figure 4.1: Other applications related to interactive topology optimization.

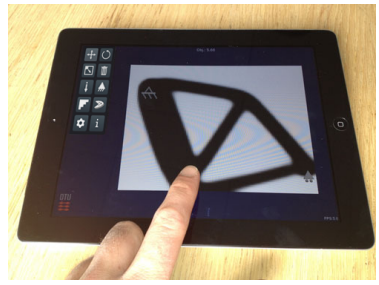


Figure 4.2: Example usage of the 2D TopOpt app: The user is moving a load while getting continuous visual feedback on how the movement affects the optimization of the structure.

feedback loop, often under one second, the user gets the feeling that he is in control of the program and the interactivity helps him understanding consequence of his actions. This relates to the user interface guidelines being both user supportive and informative as discussed in section 2.5. An example of using the app is found in figure 4.2.

The TopOpt 2D is created for both web browsers (using the Unity plugin), iOS and Android. The application was originally released in 2012, but has frequently been updated with new features, improvements and bug-fixes. The application is available free of charge at `http://www.topopt.dtu.dk`.

4.2.1 Problem formulation

The optimization problem is the same as described in equation 3.14. In addition the application should maintain a stable frame rate and the solution should be able to converge from any given design.

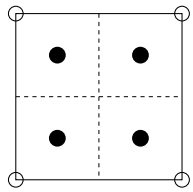


Figure 4.3: The multiresolution approach (MTOP), where the circles represent displacement nodes, the dots is the design variables (element densities), which are separated by the dashed lines.

SIMP has been used with a high lower bound on both the density and the stiffness. The stiffness interpolation is given as

$$E(x_i) = 0.01 + 0.99x_i^p \tag{4.1}$$

where $p = 3$ is the penalization factor. Using the high lower bound on the stiffness $E_{min} = 0.01$ allows the shape to easily adapt from one optimum to the next when the model is changed. The same is true for the lower bound on densities $x_{min} = 0.01$. Both E_{min} and x_{min} are gradually lowered to 10^{-7} after the optimization problem has remained unchanged for 7 iterations, but is reset to 0.01 as soon any parameter is changed. These values have been found by numerous numerical experiments.

The multi-resolution design representation (MTOP) [NPSL10] is used to additionally increase performance by obtaining a finer design representation with little extra computational cost. This is achieved by dividing every finite element into a number, in this case four, design variables as illustrated in figure 4.3. Using the MTOP approach, the element matrix contribution can be given as:

$$\mathbf{K}_e(x_e) = \sum_{i=1}^{4} \mathbf{K}_0^i x_e^i \tag{4.2}$$

where $\mathbf{K}_0^i$ is the reference stiffness matrix evaluated at each of the four design variable locations. Using MTOP has the benefit that even though the assembly takes four times as long, the size of the linear systems to be solved remains the same. Since the mesh is regular, the numerical integration of the four submatrices can be found once and be reused for all finite elements throughout the iteration process.

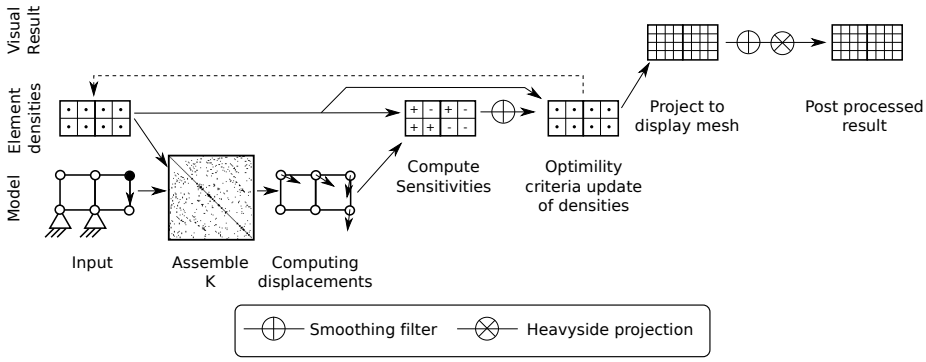


Figure 4.4: Main steps of one iteration of topology optimization in the TopOpt 2D app

To avoid the checkerboard problem, the sensitivities are filtered using equation 3.23. Since the checkerboard problem depends on the standard element discretization, the filtering must be performed with a radius relative to the physical element size. The chosen distance is set to $r_{min} = 2.6$ times the design element size (i.e. 1.3 times the finite element size).

The design update is using the optimality criterial method as described in section 3.3.5.

To improve the visual appearance the shape is refined in a post processing step before displayed. This is done by projecting the fine element densities to a display mesh with twice the resolution. To make mesh appear smooth, the display mesh is then filtered using a standard density filter with a radius of two display mesh elements. Finally, in order to sharpen the edges, a Heavyside step function is applied to the densities as described in [WLS11]:

$$\tilde{x_i} = \frac{\tanh(\beta\eta) + \tanh(\beta(x_i - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \tag{4.3}$$

with the sharpness control parameter set to $\beta = 6$ and cut-off point set to $\eta = 0.5$.

A full conceptual iteration is summarized in figure 4.4.

4.2.2 Implementation

The initial version of the app was implemented purely in Unity using C#, which allowed using the same code for both web browsers, iOS and Android apps. The cost of this portability is performance, since highly optimized linear algebra libraries such as BLAS and LAPACK cannot be used.

In the initial version we had implemented a simple sparse linear algebra library used for both filtering and for the linear solver. We also experimented with iterative solvers and the multigrid preconditioned conjugate gradient method in particular. Even through this gave a significant speedup we found the result was often less visually appealing. When we changed the optimization settings using the iterative solver, the gradient would sometimes be too inaccurate, which resulted in material being added or removed at seemingly random locations. Due to this, the iterative solvers was discarded. We also experimented with banded matrices as well as using some existing C# linear algebra libraries, however we found that our own simple sparse library was the most efficient for this use case.

Later, the optimization kernel was rewritten in native C++ code used on the iOS platform which gave a significant speedup of approximately factor 10. This performance gain was both due to changing to a more low-level programming language as well as using the SuiteSparse, BLAS and LAPACK libraries for linear algebra operations. The SuiteSparse library contains the CHOLMOD library which is used for an efficient sparse Cholesky factorization of the symmetric positive global stiffness matrix $\mathbf{K}$. This factorization is used for efficiently solving the FEM linear system. In order use a simple and MATLAB-like matrix syntax instead of the complicated C-function calls, we created OOCholmod which is released under the GPL open source license at `https://github.com/mortennobel/OOCholmod`. Matrix access and operations for both dense and sparse matrices are primarily using C++ operator overload. The library also abstracts away the two data structures used by SuiteSparse: the triplet form (column, row, value) and the compressed sparse column form (CSC). Instead, the elements of a matrix are always accessed using row and column index. When the structure of a sparse matrix has been defined a build function is invoked, which translates the internal representation from triplet form to CSC. Afterwards the library still allows efficient access to the matrix values which internally searches the CSC using bisection.

The C++ kernel uses parallel assembly of the global stiffness matrix $\mathbf{K}$ which is one of the performance bottlenecks. In order to avoid race conditions the assembly is divided into two parts, where the first part assembles nodal relations belonging to every second column of elements. When the first part is finished, then the remaining columns are assembled. Each of the spawned threads in

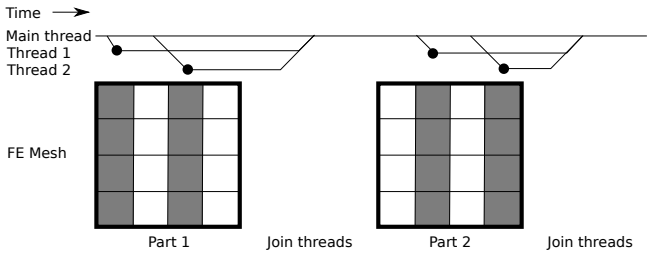


Figure 4.5: Parallel assembly of the global stiffness matrix. Race conditions are avoided by dividing the assembling into two parts, where each thread will never conflict with other threads.

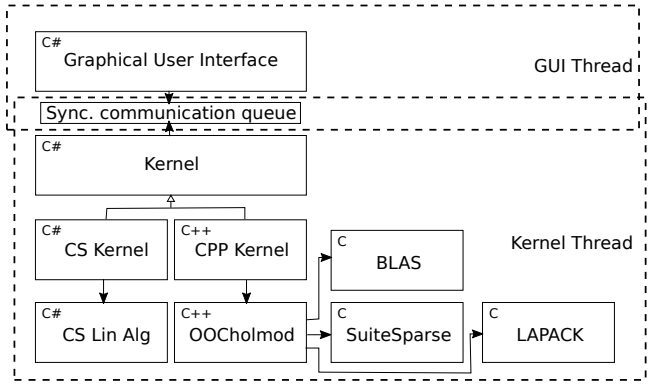


Figure 4.6: Conceptual software architecture of TopOpt App shown in UML.

the assembly is responsible for assembling a number of columns. Using this scheme the global stiffness matrix will be assembled in parallel with no risk of simultaneously accessing the same element from multiple threads. The only locking used is waiting for the first and second part of the algorithm to finish. The real benefit of using this approach comes from using OOCholmod, which provides efficient access to the values of a built sparse matrix. The parallel assembly is illustrated in figure 4.5.

In order to prevent the optimization kernel to stall the responsiveness of the user interface, the kernel runs in its own thread. In each frame the user interface thread will send any model changes to the kernel and update the visualization if an update is available. The communication between the user interface and the kernel is performed using two synchronized queues. Except for the two synchronized queues no memory is shared between the two threads. The architecture of the kernel is illustrated in figure 4.6.

4.2.3 User interaction

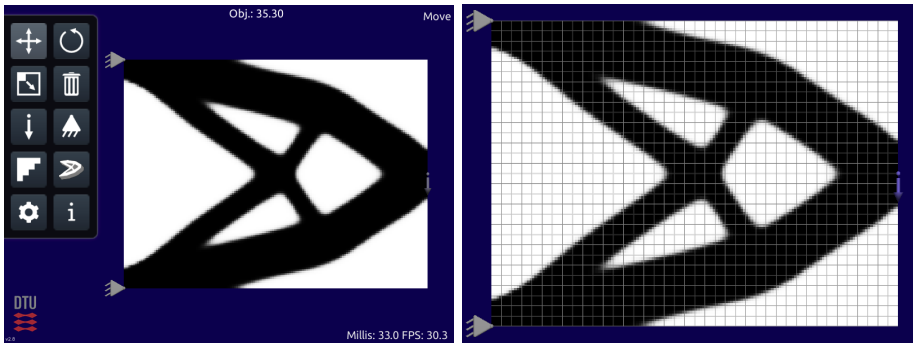
The user interface is designed for users already familiar with mechanical engineering and the symbols used in the field. Unlike other applications, where the modeling is spatially separated from the result, the users instead models the design problem on top of the optimized shape using loads, sliding/fixed supports and passive elements. This approach makes it much easier to understand the relationship between the modeling and its effects on the optimized shape.

The design strives for being informative by only displaying information when needed. For instance, the user only sees the result on the fine display grid, except when performing modeling operations where the coarse grid appears. The modeling operations also snap to the coarse grid to enforce a valid model. An icon based menu is used for selecting the current action as shown in figure 4.7. To keep the menu simple, some of its items contains sub-menus, which slides into the screen on activation. Additional information about the current action is displayed in the upper right corner, e.g. "Move (20,30)".

The design problem is modeled using a drag gesture based scheme where components can be moved, rotated or distributed. Rotation is clamped to steps of 10 degrees for loads and 90 degrees for supports. Even through rotation of supports only has effect on sliding supports, it is still possible on all types of supports to improve the visual appearance of the modeling.

Loads and supports can easily be changed to affect multiple nodes using the distribute action. Loads can be distributed horizontally or vertically, which transforms the load from a point load to a distributed load. When distributing a load, its force is distributed among the affected nodes with the end nodes weighted half of the center nodes. When loads are distributed, materials are automatically added to the elements next to the load in order to automatically enforce a more physically plausible model. Supports can be distributed both horizontally, vertically and to a rectangular region.

The application supports up to three different load cases. When using multiple load cases, each load case has its own load vector $\mathbf{f}_i$ and a displacement vector $\mathbf{u}_i$ is found for each load case. The objective function is changed to the sum of the compliances for all load cases. The load case action is used to toggle a load between three possible load cases, which are color-coded using gray, red and green. The magnitude of each load can also be changed. Finally, handheld devices includes a gravity load, where the load direction is determined by the gravitational force applied to the actual device. This will change the load direction depending on how the device is held, which couples the optimization with the real world.



(a) User interface **(b)** Coarse grid appears when modeling

Figure 4.7: Graphical user interface in the TopOpt App

The application also supports passive elements, which are elements where the densities are forced to be either void or full material, and hence left out of the optimization. Adding passive void elements can be seen as modifying the design domain, such that it is no longer a rectangular shape. Adding passive material elements forces the optimizer to have some region of the shape predefined. Passive elements are defined on the coarse mesh using a simple drawing gesture. To simplify marking a larger region as passive elements, the user can draw a boundary of a shape and its interior will be filled automatically. When the draw gesture is initiated on an existing passive element, the gesture will act as an eraser by deleting passive elements under the stroke.

The app also supports exporting the optimized shape as a 3D geometry with a specified thickness, which makes it easy to manufacture using a 3D printer. The export also allows specifying symmetries on the four sides of the design domain. The 3D geometry is created by first extracting polygons from the display element density grid using marching squares and then extruding the shape to the given thickness. Finally, the 3D geometry is sent as a STL-file to a user provided email-address. This feature uses a server-side part which generates the geometry file and sends out the email.

Other features of the app includes changing the volume fraction and penalization, restarting the optimization (which may end up in another local optimum) and changing the grid resolution.

The design of user interface has been evaluated in an informal way using both heuristic evaluation as well as user based evaluation, where test-subjects, primarily members of the TopOpt-group, was asked to try out the new app while being observed (both usability evaluation methods were discussed in section 2.3).

4.2.4 Flexible void

A recent extension to the App is support for flexible void. A flexible void region can be reshaped by the optimizer to improve the objective function while maintaining its volume as described in [CAS14]. The motivation for using this extension is to be able to solve design problems with some geometric restrictions, where the shape must include predefined void regions, due to manufacturing or functional requirements. The shape of these predefined void regions is allowed to be changed by the optimizer in order to improve the objective function. An example of this is when modeling a wall with a windows, where we have some requirements of the window size but not on its shape.

From a usability perspective a flexible void region is used in the same way as passive elements, except that after a flexible void region has been defined by the user it will be deformed by the optimizer. During editing of flexible void regions the optimization of the regions is turned off.

Due to the computational complexity, the extension is currently only available on iOS, which is the only platform with C++ and OOCholmod support. The extension was implemented by Nis Peter Lange.

4.2.5 Examples

Figure 4.8 shows how a problem of multiple load-cases is interactively modeled on a very fine mesh (50×100 elements on the coarse mesh - 200×400 on the display mesh). Note that except for the first and the last image, the images shows shapes that have not yet converged. Modeling of the design problem is performed while the optimizer is running.

The flexible void usage is demonstrated in figure 4.9. Notice how the flexible void boundary is drawn and then automatically filled. After the flexible void is inserted it is reshaped while maintaining its volume, such that the compliance of its enclosing figure is minimized.

4.3 Finger Finite Element Method

Finger Finite Element Method (FFEM) is an app created by the DTU Master students Søren Madsen and Nis Peter Lange under supervision from Niels Aage

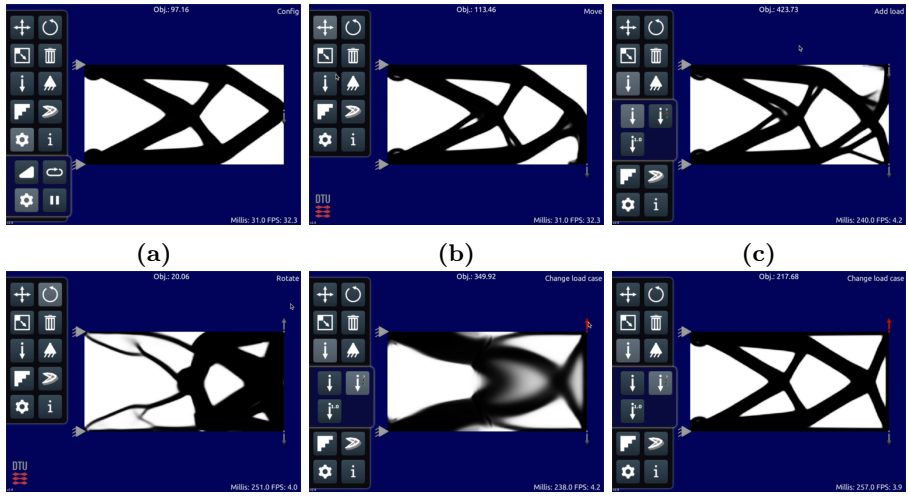


Figure 4.8: Example usage of optimizing multiple load-cases in TopOpt App. (b) Move load to lower right corner (c) Insert new load in upper right corner (d) Rotate inserted load upwards (e) Change load-case of inserted load (f) Converged result

and Morten Nobel-Jørgensen. The app is based on the code of the TopOpt app and includes some of the same user interface elements and features.

FFEM allows the user to model a FEM problem by first creating a design domain by drawing where the square elements should be located. The design domain are then augmented using fixed or sliding loads, similarly to TopOpt App. Finally, the user can touch the structure and make a drag gesture to set a prescribed displacement. Multi-touch is supported, such that many simultaneous prescribed displacements can be defined. The program will interactively show the deformed model as well as visualize the stress in the shape. An example usage of the app can be seen in figure 4.10.

The displacements and stresses are found by solving the 2D linear elasticity FE problem for plane stress as described in section 3.2. The isotropic, linear elastic material is intentionally made very soft with Youngs Modulus $E = 1$ and Poisson's ratio $v = 0.4$. The Jet color scheme is used for visualization of both Von Mises stresses, normal stresses in x or y direction, shear stresses and magnitude of displacements.

The FFEM app is freely available for both iOS and Android at `http://www.`

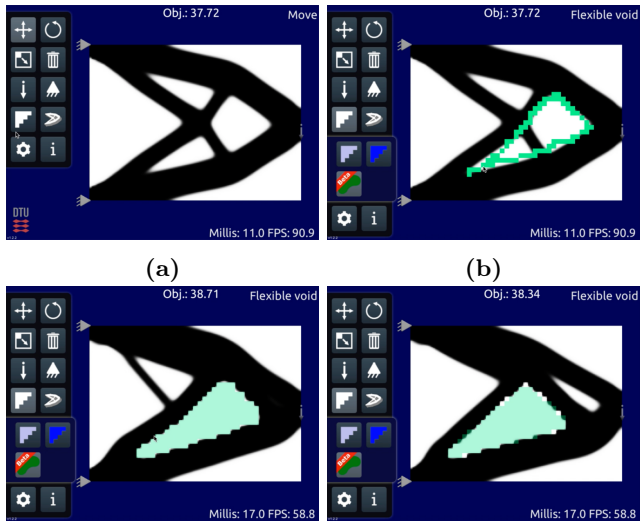


Figure 4.9: Example of using a flexible void. (b) A flexible void boundary is drawn (c) The flexible void region is filled and optimizer start optimizing both shape and flexible void region (d) The converged shape.

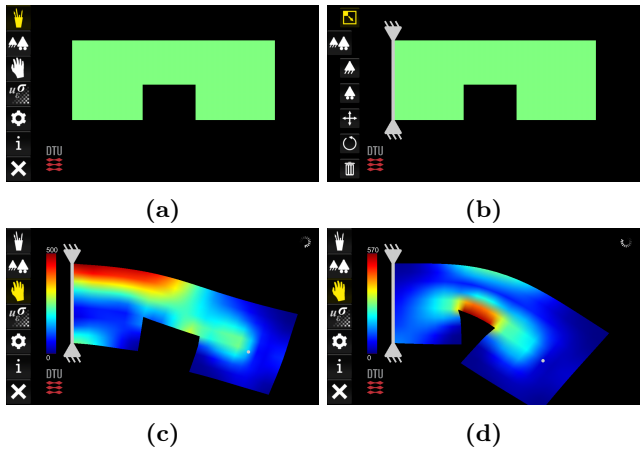


Figure 4.10: Example usage of FFEM, where (a) shows drawing the design domain (b) supporting the designed domain using fixed supports (c) and (d) shows deformation and visualization of Von Mises stresses.

topopt.dtu.dk/?q=node/891.

4.4 TopOpt 3D

The TopOpt 3D app was created as a successor to the TopOpt 2D App. The goal of the app is to extend the functionality of the TopOpt 2D app into 3D while maintaining a simple and intuitive user interface.

4.4.1 Problem formulation

The topology optimization problem solved in TopOpt 3D is the same as equation 3.14, with the exception that the modified SIMP is used, which allows element densities to become zero:

$$\begin{aligned} \min_{\mathbf{x}\in R^n} \quad &: \quad c(\mathbf{x}) = \mathbf{u}^T\mathbf{K}(\mathbf{x})\mathbf{u} && \text{(Compliance)} \\ \text{subject to} \quad &: \quad \mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f} && \text{(FEM equilibrium)} \\ &\quad \textstyle\sum_{e=1}^{n} \frac{\mathbf{x}_e}{n} \leq v_{max} && \text{(Volume)} \\ &\quad 0 \leq \mathbf{x}_e \leq 1 \quad e = 1, \ldots, n && \text{(Design variables)} \end{aligned} \tag{4.4}$$

4.4.2 Implementation

Going from a 2D FEM problem to a 3D FEM problem is a huge increase in complexity, e.g. the local stiffness matrix, describing the stress/strain relations of a single element, grows from $(4 \times 2)^2$ to $(8 \times 3)^2$ for bilinear square and trilinear cubic elements. One consequence of this is that the global stiffness matrix becomes much larger for the same number of elements.

To achieve interactive update rates on hand-held devices, the optimization kernel used in TopOpt App 3D is implemented in C++ using the libraries OOCholmod, SuiteSparse, LAPACK and BLAS.

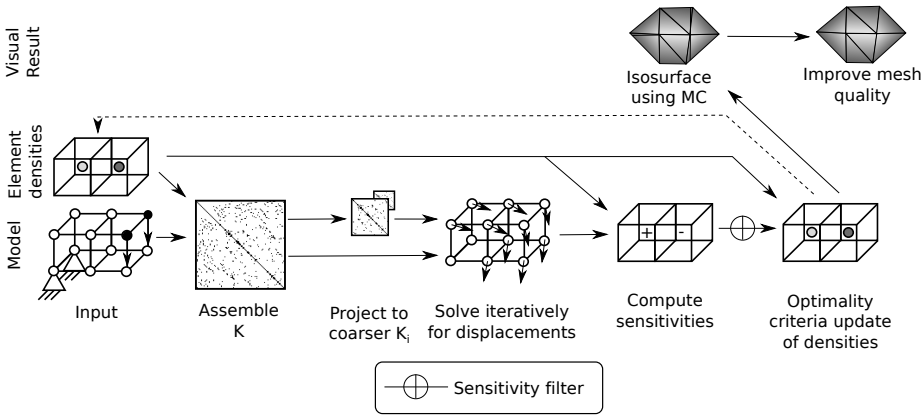


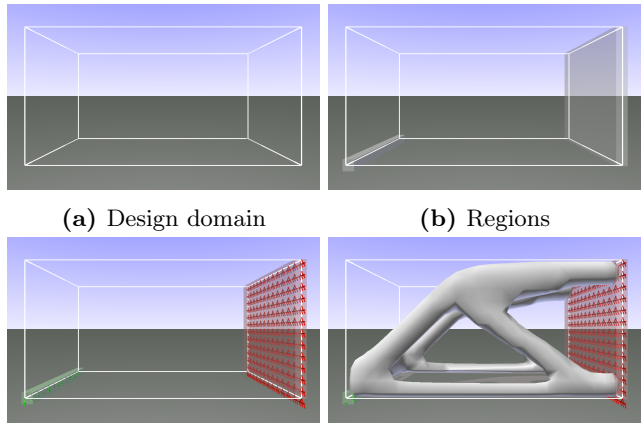
Figure 4.11: Main steps of one iteration of topology optimization in the TopOpt 3D app. First the global stiffness matrix $\mathbf{K}$ is assembled using the element densities, the design model as well as the local stiffness matrix $\mathbf{K_e}$. $\mathbf{K}$ is then projected to coarser matrices, which are used in the iterative solver to find the displacements. The sensitivities and the objective value are then found and a filtered sensitivity field is used to update the element densities using the OC method. The updated element densities are used for extracting the isosurface using marching cubes algorithm. The mesh of the isosurface is finally optimized to improve the visualization.

For penalization the modified SIMP scheme is used with a penalization value of $p = 3.0$.

In contrast to TopOpt App 2D, an iterative solver is here used to solve the FEM problem sufficiently fast. The solver used is the geometric multigrid preconditioned conjugate gradient (MG-PCG) method described in [AAL14]. We use the solver without the premature termination heuristics presented in [AAL14], and instead use a default of three multigrid levels and a relative tolerance of 10^{-5} as a convergence criteria for the linear solver.

During each iteration a number of checks is made to ensure validity of the design problem. These include that the FEM system is non-singular, etc. In the rare case that a problem is detected, the user is informed using an error message and the optimization is paused until the problem has been resolved.

The sensitivities are smoothed with a filter radius of 1.6 in order to ensure existence of the solution and to avoid checkerboard problems.



(a) Design domain **(b)** Regions

(c) Augmenting **(d)** Optimizations

Figure 4.12: Concepts of the the TopOpt 3D.

As in the 2D app we use a high lower bound on the stiffness $E_{min} = 10^{-4}$ which after 7 iterations are gradually lowered to $E_{min} = 10^{-7}$.

Finally, in order achieve a good visualization of the result, an isosurface mesh is extracted from the element density field using marching cubes with a threshold of 0.5. The resulting mesh is further refined by removing thin needles and minimizing the curvature energy using an edge-flipping, greedy algorithm [DHKL01]. Finally, the vertex normals are found using the angle-weighted pseudonormal scheme [BA05].

Figure 4.11 outlines a single iteration of the optimization.

4.4.3 User interaction

The user interaction of the TopOpt 3D is very similar to the TopOpt 2D. Figure 4.12 shows the main concepts used in the application. In a design domain (4.12a) volumetric regions (4.12b) are used to define loads, supports and passive elements (4.12c). The optimizer will continuously update the shape (4.12d) to minimize the objective function such that the modeling of the problems becomes an exploration of optimized shapes.

The user interface design of the TopOpt 3D app based on three main principles: Simplicity, continuity and consistency. These principles are a subset of list of principles described in [Tog14] and also relates strongly the rules described in

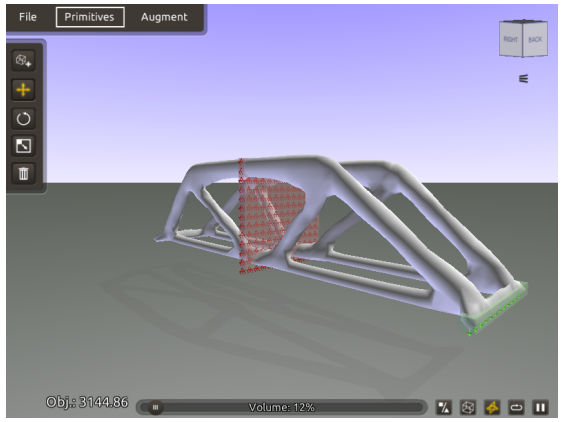


Figure 4.13: The user interface of TopOpt 3D app

section 2.5. The user interface of the application is shown in figure 4.13.

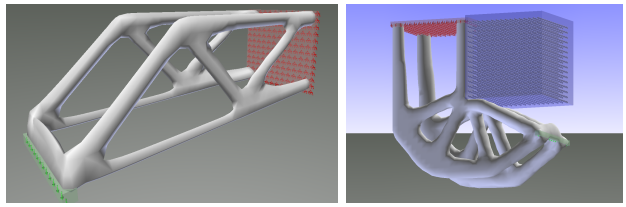
Simplicity principle: "Everything should be made as simple as possible, but not simpler." This famous Albert Einstein quote is valid both for physics as well as user interaction design. When designing applications to be run on the small screen of todays smartphones, the physical screen size combined with the inaccurate touch input gives additional restrictions on the amount of information and choices presented to the user at a given time. To keep the user interface simple and compact, the available actions are mainly displayed using icon buttons.

The application uses a single viewport for the 3D content, which shows both the model problem as well as the optimized shape. This is in contrast to the use of multiple viewport commonly found in other programs for manipulating 3D content.

Continuity principle: In order to build the users mental model of the system, it is important to visually show how the actions are performed in a continuous way instead of using instant transitions between the system states.

One example of the continuity principle is found when modeling the problem; Loads, supports and passive elements are specified using augmented regions (such as cubes or spheres). When a new region is inserted into the design domain it is animated as if it was falling from the sky. This allows the user to see the effect of the action as well as understanding where the new region is located - even if the destination is partial or fully occluded.

Another important usage of the continuity principle is found in the ViewCube



(a) Cantilever **(b)** L-shaped cantilever

Figure 4.14: Standard examples used in topology optimization.

widget located in the upper right corner. The ViewCube widget, invented by Autodesk [KMF+08], allows the user to easily see the orientation of the camera relative to the shape using the named sides of the cube. The ViewCube can also be used to change the camera view by either dragging the cube or by clicking on its face, edge or corner. This will rotate the view to the desired destination using a short, smooth animation. Gradient based horizon and shadows projected to the ground are two other visual cues used for increasing the users perception of both the camera orientation and the shape.

Consistency principle: By having consistency in the user interaction, the user can learn a concept once and apply the same concept in many different situations.

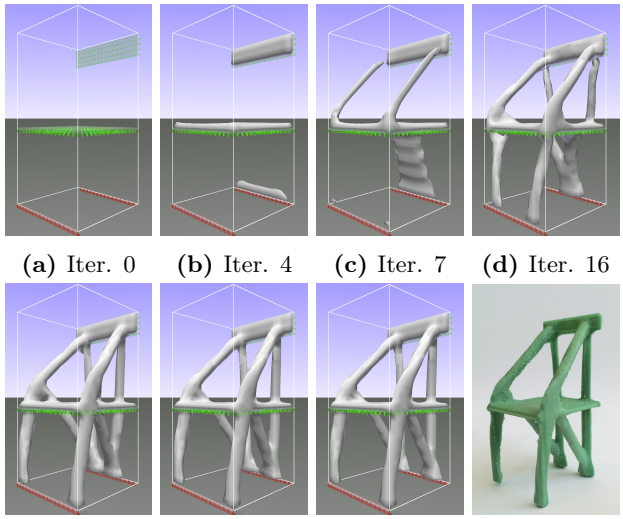
Consistency is for instance found in the way that the regions in the design domain are transformed: When the user has selected a region, using click or lasso selection, the region can be moved, rotated or scaled using the 3D widget appearing at the pivot point of the region.

Additionally, the application allows the user to load or save the design model as well as exporting the optimized shape as an OBJ file.

4.4.4 Examples

Figure 4.14 shows two of the standard examples used in topology optimization: a cantilever beam and an L-shaped cantilever.

Figure 4.15 shows how a chair can be modeled. The ground is modeled as support using two bars, the seat using a plane with vertical loads and the back-rest using a plane with backwards-pointing loads. After only 16 iterations the shape begin to look like a chair. After approximately 150 iterations the topology of the chair changes from a tree-legged chair into a four-legged chair. The whole



(a) Iter. 0 **(b)** Iter. 4 **(c)** Iter. 7 **(d)** Iter. 16

(e) Iter. 100 **(f)** Iter. 150 **(g)** Iter. 200 **(h)** 3D print

Figure 4.15: Modeling a chair (volume fraction 12%).

process of modeling and optimizing the chair only takes around 2-3 minutes. If the shape does not satisfy the user during the modeling / optimization, the user is able to guide the shape at any given time by changing the design model. Often this can be done by adding passive void regions to force material away from certain areas in the design domain. The last image in the sequence shows how a shape can be exported and manufactured using a 3D printer.

4.5 Discussion

The applications presented in this chapter have all been enormously well-received both in terms of download statistics and user reviews as shown in Appendix E. We see this as a validation of both good performance, well-chosen features and well-designed usability. Most successful is the TopOpt app which in a little over three years has been downloaded over 21,000 times. This clearly surpasses its predecessor which had 13,000 users over a period of 13 years.

The predecessor used a client-server solution with a thin Java Applet front-end and server back-end for performing the computations. This setup has the benefit of performing equally well on all platforms, only limited by the server capabilities and the network connection. Instead of continuing this approach, we

chose a purely client based solution as the compute powers of modern hand-held devices made this possible. This approach has the benefit of a much simpler software architecture and does not rely on network connections or servers.

Another choice is whether to perform the computations only on the CPU or also use the GPU as a co-processor (GPGPU). Even though GPGPU is an obvious choice when solving computationally demanding problems, we chose not to use this for the following reasons: While the variation in performance on PC and mobile CPUs varies around a factor of 10, the variation of the capabilities of GPUs is much higher. Besides the mobile GPU architecture is often less suitable for general purpose computations. GPGPU APIs are also not yet common on mobile platforms.

A final choice is software platform. We decided to use Unity due to its good support for cross-platform development. If the projects were to be recreated today, one alternative to consider is to target a pure browser-based solution in contrast to Unity's browser plugin. This means that the browser is executing code defined in JavaScript, asm.js (a high performance subset of JavaScript) or the recently announced WebAssembly (low-level browser bytecode with support for SIMD instructions) [Bri15]. Currently, pure browser performance is approximately four times slower than native code [Ben15], but will be much more portable than native or plugin based solutions. These performance issues could be solved by going back to a client-server architecture.

Chapter 5

Gamification of topology optimization

In order use topology optimization to its full potential, we find it important to have a good understanding and intuition about how the method behaves. This is especially important to possess to avoid just using Topology optimization as a black-box solver but instead being able to question its results and adjust its parameters appropriately to fit ones needs. This includes problems where the topology optimization method gets stuck in a local optimum and optimizations where the design model contains errors leading to an unwanted result.

Education is another place where intuition about topology optimization is important. Intuition is built from both knowledge and experience, hence it is important that education both contains theory as well as hands-on experience. When working with complicated topics such as topology optimization, it is easy for students to get lost in the many details of the problem. This can hinder a high-level understanding of the nature of the problem and its solution. A related aspect is the importance of repetitions as a technique to reinforce the knowledge.

This chapter describes the TopOpt Game and how this game has been designed for training the intuition of topology optimization. In the game the player solves various topology optimization challenges in a number of levels. The chapter will

first highlight some of the related work, then discuss the game and finally make a statistical analysis of the gameplay data to unveil the improvements gained from playing the game. The TopOpt Game is also the topic of Paper C.

5.1 Related work

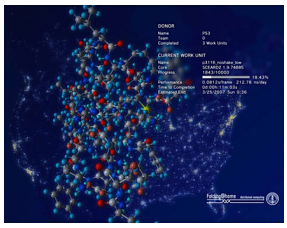
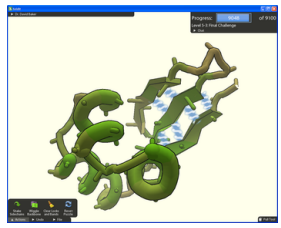
Gamification, the use of game elements added to non-game context, has been used several places, including science. One of the most successful scientific applications of gamification is within research of protein folding. Computing how proteins fold is a hard non-convex optimization problem and is an important research field in order to understand and potentially cure many diseases such as Alzheimer's, Parkinson's and some types of cancer. The following two research projects use gamification for protein folding (also shown in figure 5.1a and 5.1b):

- **Folding@Home** allows people to donate CPU time of their computer when it is not used. The project currently has over 140,000 computers associated which act as a distributed cluster with the total compute power of over 30,000 teraflops. Folding@Home is using gamification by having leaderboards, which motivates people to compete, individually or in teams, by donating CPU resources.

- **FoldIt** has transformed the problem into a puzzle game by using meaningful and simple abstractions. In FoldIt the players search for optimal solutions for protein folding problems. The game starts with a tutorial containing simple problems and a limited set of tools. The player is challenged by a gradually increasing complexity and by more complex tools. In some cases, the performance of the players has beaten existing protein folding algorithms. User behavior in FoldIt has been studied in order to extract strategies to improve the protein folding algorithms.

Besides the problem solving aspect, the two projects also have the positive side effect of increasing peoples awareness of the diseases related to protein folding.

Other related work includes games where mechanical engineering is used as a game mechanic. A good example is found in Bridge Constructor by Headup Games, where the goal is to design a bridge using truss structures such that the bridge can withstand loads from crossing cars and trucks as shown in figure 5.1c.


(a) Folding@Home **(b)** FoldIt **(c)** Bridge Constructor

Figure 5.1: Related work. (a) and (b) is examples of gamifications used in science and (c) is a game where mechanical engineering is the core game mechanic.

5.2 Problem formulation

TopOpt Game is based on the discrete problem formulation from equation 3.13. However, to follow the conventions used in games, the problem is reposed as an optimization problem where the objective function is to maximize the score:

$$
\begin{aligned}
\max_{\mathbf{x}\in R^n} \quad &: \quad s(\mathbf{x}) = \tfrac{10^8}{c(\mathbf{x})} && \text{(Score)} \\
\text{subject to} \quad &: \quad c(\mathbf{x}) = \mathbf{u}^T\mathbf{K}(\mathbf{x})\mathbf{u} && \text{(Compliance)} \\
&\quad\;\; \mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f} && \text{(FEM equilibrium)} \\
&\quad\;\; \textstyle\sum_{e=1}^{n} \frac{\mathbf{x}_e}{n} \leq v_{max} && \text{(Volume)} \\
&\quad\;\; \mathbf{x}_e \in \{x_{min}, 1\} \quad e = 1, \ldots, n && \text{(Design variables)}
\end{aligned}
\tag{5.1}
$$

where x_{min} is the lower bound on the design variable.

The player of the game acts as the optimizer trying to optimize the score by modifying the design variables. That is the player will distribute material within a design domain in a given level in order to find a good solution for the problem posed.

As mentioned in section 3.3 this discrete topology optimization is in general not feasible to solve, which means that the maximum score for a given level is

unknown. To be able to rate the performance of the player a baseline score is approximated using the density approach.

The maximum score varies between levels since it depends on the problem posed. It is also important to realize that the relationship between the player's score of a given level and his "performance" is not linear. A valid, trivial solution (e.g. by connecting all loads with supports using available material) often gives around 50% of the estimated maximum score, whereas finding solutions close to the approximated maximum score is much more difficult.

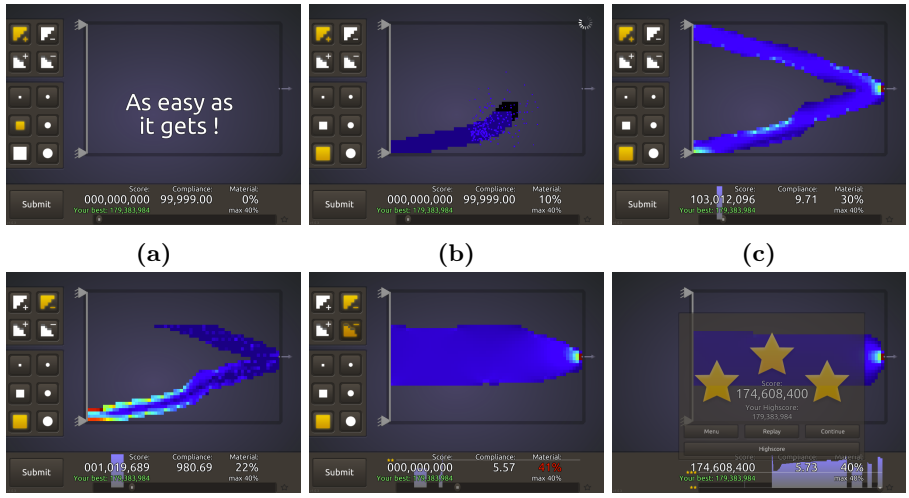
5.3 Game design and implementation

The TopOpt Game has been designed as a puzzle game - a genre of games where the player is challenged by problems with increasing difficulty and given rewards when progress is made. In the TopOpt Game each level challenges the player with a topology optimization problem composed of a design domain, loads, supports and a limit on the material. The goal of the player is to find the optimal material distribution, which will give the highest score and minimum compliance. To keep a steady flow of challenges and to force the player to be focused each level must be completed within a time limit between 30 and 180 seconds. As a visual hint the strain energy density is displayed on the structure using the Jet color scheme, allowing the player to easily identify problematic areas with high strain energy density.

A typical gameplay is presented in figure 5.2, where the player first tries one strategy (5.2a - 5.2c), then erases the design (5.2d) and finally finds a better solution (5.2e - 5.2f). During the gameplay the player uses too much material (5.2e) which is penalized by setting the score to 0. To give the player a final feedback on his performance the player is rewarded with 0-3 stars based on how the achieved score relates to the predefined baseline score.

The player distributes material in the design domain using brush stroke gestures similar to a painting program. There are six different brushes available in various shapes and sizes. Four different tools are used to alter the material distribution; One for adding, one for removing and additional two tools which add or remove elements on the boundary of the shape.

One of the important task of the user interface of the game is to provide feedback to the player about his current performance. This is done using the score label and the score graph as shown in figure 5.3. The score graph shows the progression of the score over time, which makes it easy for the player to see how



(a) (b) (c)

(d) (e) (f)

Figure 5.2: Gameplay: While the user paints a solution the game provides feedback in terms of a score and by visualizing the strain energy density using the Jet color scheme.

Figure 5.3: Score graph: Sliding the handle left or right allows easy backtracking. The green text shows the personal high-score from previous games.

his actions affects the score. This reduces the players short-term memory load, as described in section 2.1.2, by not having to remember previous scores. The score graph also allows to easily backtrack to a previous design by sliding the handle below the graph to a previous time point. Finally, the score graph also acts as a countdown timer, such that when the graph reaches the right side then the time is up.

The game is designed for players to primarily compete against themselves. The game keeps track of the maximum score and number of stars each player has achieved in each level. This player highscore is highlighted both during the gameplay as well as in the award popup in the end of each level. On iOS the game is additionally using GameCenter. This has the benefit of viewing stats

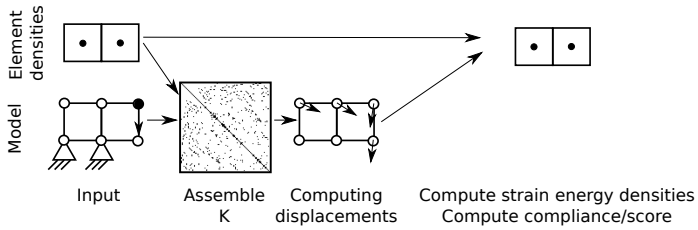


Figure 5.4: Evaluating the score in the TopOpt Game.

on friends progress as well as being able to send challenges to friends in the case that a good solution for a level has been found.

Performance is an important part of the application. Whenever the player has changed the material distribution it is important to give him feedback, in terms of a score and updated strain energy visualization, within a reasonable time, e.g. a few seconds. A too large delay between the action and the received feedback will make the game feel non-interactive and will disturb the gameplay flow.

The architecture of the game is similar to TopOpt 2D as shown in figure 4.6. The evaluation of the score and compliance is computed in its own thread. This allows the player to continue working on the design, e.g. using a draw gesture, even through the last material distribution has not been evaluated. If an update occurs when an unprocessed updated is waiting to be evaluated by the solver thread, the new update will then overwrite the unprocessed update. This means that only the newest updates are evaluated and some old updates may be discarded. The kernel is also written in C++ using the libraries OOCholmod, SuiteSparse, LAPACK and BLAS.

The C++ kernel is simpler than in the TopOpt app, since the game does not need to compute the sensitivities or to perform the optimization. Figure 5.4 shows how the score, compliance and strain energy density is computed. First the global stiffness matrix is assembled using the players material distribution and the design model. The FEM linear system is solved to find the displacements and finally the compliance, the score and the strain energy field can be computed. In contrast the TopOpt 2D, the MTOP is not used, since this method does not penalize solutions that are not watertight. This had the implication that solutions where the material was separated by an empty row or column on the fine grid were evaluated as if they were connected. Not using MTOP means that the full global stiffness matrix must be assembled and solved which is significantly slower that using MTOP. To keep a fast score evaluation the resolution of the design domain is therefor limited to a maximum of 60×60 elements.

The previously mentioned apps described in chapter 4 were primarily designed as standalone apps. In contrast, the TopOpt Game is designed as a connected app where Internet availability is assumed. This means that the player optionally can login and share the account between multiple devices or alternatively play using an anonymous account. All game data, such as levels and highscores, are stored on a server using the Google App Engine. To be able to analyze the player performance, all gameplay data is also stored on the server. The gameplay data contains all information about the player performance during gameplay including every evaluated action the player has performed and its associated score and compliance. The game also supports playing offline, where the data gathered is sent next time the game connects to the server.

5.4 Level design

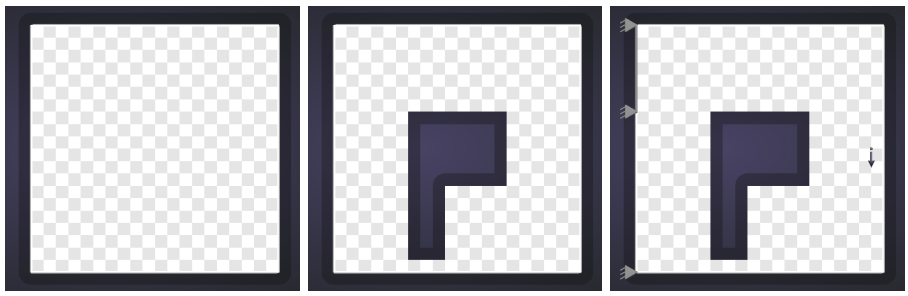
One problem in creating a topology optimization game is how to generate levels, which gradually increases in complexity and still remains fun and challenging to play. Procedurally generated levels would be one way to go, however to come up with an algorithm for this would be a very hard problem to solve. Instead the levels in the game are crafted manually in the built-in level designer.

The level designer essentially uses the same user interface as the TopOpt app. The tool allows the user to create a level design composed of a design domain, loads, supports and possible passive elements as shown in figure 5.5. Additionally information about the level must also be provided, such as volume fraction, available time, difficulty, etc.

The level editor is available for all users of the app. A custom designed level can be suggested as a new global level by submitting it from the level editor. When a suggested level is approved by an administrator, it is public available for everybody to use. This means that the game is not static, but will evolve when new challenges are being added.

5.5 Analyzing player performance

Since the TopOpt Game is created for educational use, it is important to evaluate the effect of playing the game. Based on actual gameplay data gathered when people have played the game, the following analysis aims to unveil the relationship between the number of levels a player has played (the experience



(a) (b) (c)

Figure 5.5: Designing a level for TopOpt Game, by (a) defining a rectangular design domain, (b) adjusting the design domain using passive void elements and (c) augmenting the design domain using load and supports.

of the player) and the score the player achieves (their intuition about topology optimization). Note that the player is allowed to play the levels in any order.

To perform this analysis the following simplifying assumptions have been made:

1. Each player account correspond to a single person
2. Players are always trying their best
3. A high score correlates to a high intuition of topology optimization

The score has been normalized in order to reduce the effect of different complexities across levels. Besides, when the player scored 0 due to violated volume, the associated gameplay data was removed from the analysis since we found that this was more related to the rules of the game than to the intuition of topology optimization.

Figure 5.6 shows the normalized scores of all observations plotted as a function of experience. The graph is clamped at 150, since the number of observations beyond that point is very low. In the figure a Linear Mixed-Effect (LME) has been applied to model the observations. In the LME the players are modeled as the random effect, since players cannot be assumed to have equal skills with regards to topology optimization prior to playing this game. Also their learning curve (increase of score as a function of experience) may vary. Figure 5.6 shows that the data is very scattered and noisy, however the red line plotting the LME model shows a correlation between the increase in experience and the increase

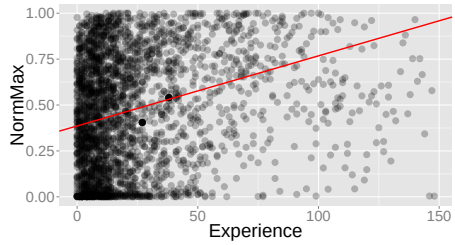


Figure 5.6: Normalized score based on experience with Linear Mixed-Effect model fit (red line).
$NormScore = 0.0038 \times Experience + 0.3861$

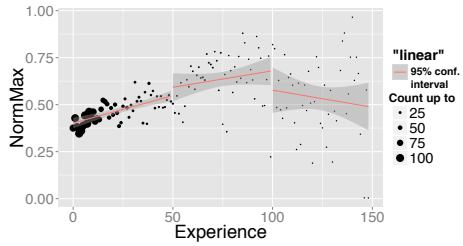


Figure 5.7: Linear Regression analysis

in score with significant parameters (< 0.001) in the model for both slope and intercept.

The same trend can be seen by using the simpler Linear Regression analysis (LR) to find the relationship between the player experience and the normalized score as shown in figure 5.7. Since the densities of the observations are poor for observations over 50, the analysis has been performed in intervals of 50. The first two intervals shows a clear relationship where experience will increase the score. The third interval shows a drop in player performance, but the data is very sparse and we believe more data is needed before any conclusions is made for this interval.

We therefor conclude that there is a statistically significant correlation between playing the game and increasing one's intuition of topology optimization. The gameplay data is however very noisy, which we believe is due to topology optimization being a hard problem for human to solve as well as our simplifying

assumptions may not always be valid.

5.6 Discussion

TopOpt Game allows players to build up a intuition about topology optimization by solving interactive puzzles. The game continuously provides feedback about the player's progress using the score graph and by visualizing the strain energy density. This creates a nice feeling of flow when playing the game. The game is also relatively fast-paced due to the limited time available for optimizing the topology within a level.

However, we are aware of a few areas where the game could be improved. By providing a better introduction to the game, in terms of for instance interactive tutorials or better documentation, the game could give a better experience for people outside its current target group of mechanical engineers. This could also be used to emphasize important learning points when new concepts are introduced (such as sliding support or multiple load cases). Ideally, the game should be able to provide more hints to the player during gameplay in order to guide the player from a trivial solution to a good solution.

In a learning situation it is probably a good idea to first introduce topology optimization using the TopOpt App. When the students have become familiar with its concepts then the game can be used to give the students a feel of the complexity of the problem.

In many ways TopOpt Game is very similar to FoldIt - both asking the players to solve very hard problems. One thing that FoldIt does particularly well is how it uses abstractions to make the problem solving more accessible. For instance it has tools such as rubber band (creates a soft constraint) and shake (randomizes the solution slightly to avoid local optimum). If similar abstractions could be found for TopOpt Game, then this could also improve the usability of the game.

Chapter 6

Rethinking topology optimization as a modeling tool

The main focus of the applications described in chapter 4 is to find solutions to the topology optimization problem in an interactive manner. These applications are mainly valuable for educational use and as playgrounds for engineers, whereas designers and architects will have some issues with using them as shape modeling tools. The main challenges these applications have from a design-tool's point of view are:

- **Lack of local control**. Often designers are interested in controlling the shape on both a global and local scale. The tools presented so far all model the shape on a global scale, maybe with the exception of passive elements which provide some local control.

- **Lack of predictability**. A related problem is that small local changes potentially end up in a significant different shape and topology, which may not be aesthetically pleasing. Since aesthetics cannot easily (if at all) be formulated mathematically as an objective function, the lack of predictability is a huge issue for designers.

- **Indirect modeling**. The interaction of the modeling is indirect, meaning that the shape of structure is changed by modifying some of the constrains or optimization parameters. This indirect form of modeling is challenging for designers who often prefer a more direct and explicit control of the modeling.
- **Lack of expressiveness**. The designers may not feel that the tools provide them with the expressiveness they need for supporting their creative process.

These challenges have strong relations to first usability design rule "user supportive" from section 2.5, since the designer will have a hard time predicting the consequences of his actions. Some of these challenges may be inherent problems in topology optimization, however we at the same time believe that part of the problem is due to traditional thinking and lack of experimental research in designing such tools.

This chapter will describe a new experimental way of using topology optimization that better supports designers in their creative work of creating optimized structures. First, the chapter will go through related works. Then follows presentations of the Deformable Simplicial Complex (DSC) method and how DSC can be used for topology optimization. Finally, it describes the proposed tool including implementation details as well as a discussing of the found results and limitations.

6.1 Related work

We are not aware of any research or commercial projects which tries to use topology optimization as a fully integral part of a design tool. Rather, topology optimization tend to be used as distinct task performed in a non-interactive manner.

However, a lot of related research is performed on interactive design of structures with physical or other constraints. Such design tools use different methods to support the designer:

Guided Exploration. Here the program allows the designer to explore the design space more or less freely. To ensure the physical constraints on the structure are fulfilled the structure is frequently analyzed and areas with problems are presented to the user along with one or more suggested solutions. This is a very attractive method that does not suffer from any of the four challenges

described above. This main challenge with the method is to find multiple meaningful alternative solutions for at given problem. For this reason the method has only been used on problems with inherent limited design space or other problems where solutions can easily be obtained.

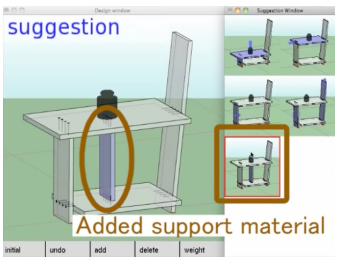
A good example of guided exploration can be found in the furniture design tool described in [UIM12]. Here the user can design nail-joined furniture, such as bookcases. During the modeling of a furniture the system will continuously evaluate the physical validity of the structure, both in terms of whether the nail joints are durable and whether the stability of the structure meets the requirements. If a problem is identified then the system will present the problem to the user along with proposed solutions. A screenshot of the tool is shown in figure 6.1a.

Sketch based modeling. Another approach is to allow abstract modeling of a structure, such as creating a sketch, and then use an algorithm to find the actual design (possibly using topology optimization). Similar to guided exploration this approach has a large design freedom. One potential problem is too large differences between the sketch (or the vision of the designer) and the actual design. Sketch based modeling is usually performed without physical constraints. The basic challenge in sketch based modeling is to deduce a complete model based on a sparse model, such as creating a 3D model based on one or more sketches created in 2D. An overview of sketch based modeling can be found in [OSSJ09].

A very popular example of sketch based modeling is Teddy [IMT99] and its successor SmoothTeddy, which transforms 2D silhouette sketches into 3D polygon surfaces. The example application of Teddy is the generation of stuffed toy animals. The user starts by creating a silhouette, which is then automatically transformed into a 3D shape by first defining a spine of the structure and then revolving the outline around that spine. From here on the user will be sketching on the 3D model. The program allows the user to rotate the model during the sketching in order to provide details at arbitrary positions in 3D using 2D sketching. A screenshot of the app is shown in figure 6.1b.

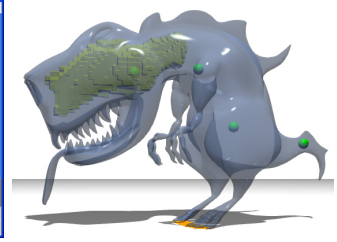
Shape optimization. Shape optimization is usually formulated as an optimization problem, where the goal is to optimize an objective by moving the surface of a shape while satisfying some constraints. When used in interactive applications, the optimization is then performed "in collaboration" with the user, for instance the user can guide the shape by interactively changing the objective function or the constraints.

Make It Stand [PWLSH13] is an excellent example of how physical problems can be solved in an interactive manner. The application solves the imbalance problems, such that a 3D model can stand after fabrication (typically 3D print-



(a) Guided Exploration of Physically Valid Shapes for Furniture Design

(b) SmoothTeddy



(c) Make It Stand

Figure 6.1: Examples of research in interactive modeling tools.

ing). Using a simple user interface the designer can provide a few control points, which are used both by the designer to alter the model, as well as the optimizer to ensure a well balanced models as shown in figure 6.1c. The optimization will try to preserve surface details by primarily doing inner carving and secondary deform the shape using the control points. A similar, but more automated, approach using shape and topology optimization is found in [CSB14].

6.2 The Deformable Simplicial Complex method

The implementation used here is based on topology optimization using the deformable simplicial complex (DSC) method ([MAB10], [MB12]).

DSC is a method for deforming shapes represented using simplices (triangles for 2D and tetrahedral for 3D), while maintaining a high quality meshing. The method has been successfully applied on fluid simulation ([MB12], [MEB+14]) and combined shape and topology optimization ([CBS], [CNJA+14], [CBNJ+15], [CTAS14]). DSC software libraries for 2D and 3D are open source under the GPL license and are available for download at `https://github.com/janba`. The following description is limited to the 3D version, however, for clarification many of the concepts are explained using 2D illustrations.

The main advantages of DSC are:

- **Explicit surface representation**. The surface is explicit defined during the deformation.

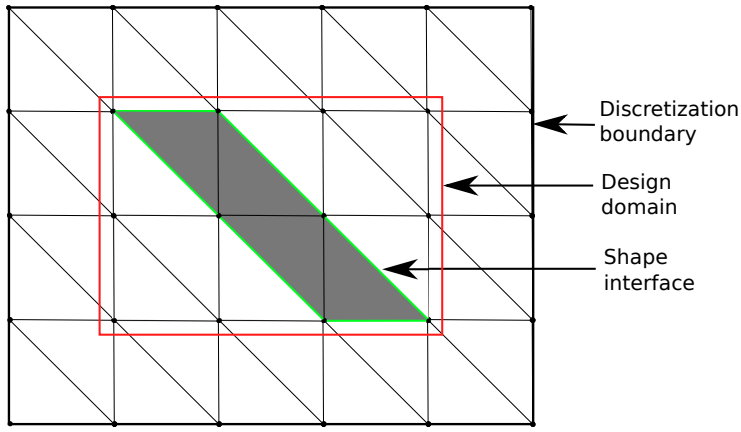


Figure 6.2: A 2D DSC discretization of a shape including a optimization domain.

- **Topological adaptivity**. Topology changes are an integral part of the method during the mesh quality optimization step.
- **Multiple optimization strategies**. The object shape can change by either moving the interface nodal position or relabeling the simplicial elements.
- **Little numerical diffusion**. Fine details are maintained during deformation.

Before starting the deformations a design domain needs to be specified. The bounds of the design domain is fixed, which means that it needs to be large enough to encapsulate both the initial shape, all intermediate shapes as well as the final shape. Since the boundary of the discretization is fixed, DSC requires some additional space between any shape boundary and the design domain. The discretization is using tetrahedrons, where each tetrahedron is initially labelled as being either material or void. An example of such discretization is shown in figure 6.2. Multiple types of materials are supported by providing a unique label for each material. In DSC the term 'interface' is used for describing the interface to the enclosed shape, whereas the term 'boundary' is used for the boundary of the discretization.

DSC is designed for iterative solutions where the surface in each iteration moves while maintaining a high quality meshing until the solution has converged (e.g. when the surface movement becomes significant small). This can be described using two steps:

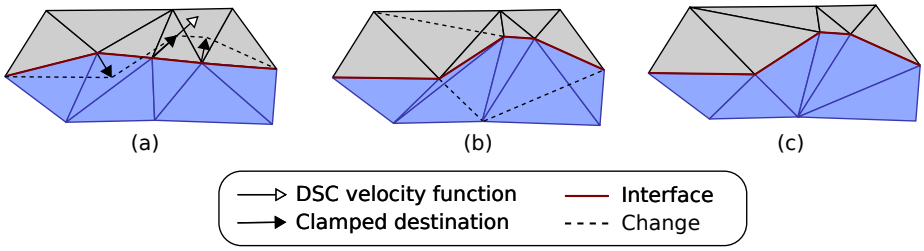


Figure 6.3: Conceptual overview of DSC in 2D. (a) Destination of the interface nodes are found by clamping the velocity function (b) after nodal movement the mesh is optimized - here using edgeflips only (c) result after one iteration.

1. **Nodal Movement**. Based on a user provided velocity function the position of each node is moved. Each movement is limited by the adjacent tetrahedrons, such that each movement will not make any tetrahedron degenerate. See figure 6.3a.

2. **Mesh optimization**. Using five local mesh operations the high quality of the mesh is maintained. See figure 6.3b.

The five local mesh operations used are:

- **Smart Laplacian smoothing** as described in [Fie88] moves the node position to its barycenters if the node is not interface and not boundary. This is only applied if the local mesh quality after movement is improved or the quality is above a user defined threshold.

- **Edge removal** is performed using a sequence of 2-3 flips followed by a single 3-2 flip as described in [She02]. The sequence of 2-3 flips which improves the local mesh quality the most is found using dynamic programming as described in [Kli80]. Edge removal can also be applied on the interface or on the boundary of the discretized domain where the final 3-2 flip is replaced by a 4-4 flip or 2-2 flip respectively. The 4-4 flips are only applied on the a sufficiently flat surface.

- **Multi-face removal**. The opposite operation of edge removal is multi-face removal as described in [She02] performed using a single 2-3 flip following by a sequence of 3-2 flips. Not applicable on interface or boundary since it would change these.

- **Node removal** merges two end nodes of an edge if this does not result

in any inverted tetrahedra. The end nodes may reside on the interface. Used to remove low quality tetrahedron when above operations fail.

- **Node insertion** inserts a new node at the barycenter of an edge. The node insertion is mainly used to increase the mesh quality in cases where other operations fail.

The heuristic strategies for using these five mesh operations to ensure high mesh quality are described in full details in ([MAB10], [MB12]).

One important consequence of the mesh optimization is that it will join two parts if the separating space is too thin. Likewise a shape can be teared into two parts if the mesh is sufficiently thin in the middle of the shape. Another important feature is that mesh optimizing operations are only used when needed, which makes the method more efficient than a complete remeshing.

The mesh detail level in DSC is mainly defined by the parameter δ corresponding to the average edge length of the discretization. The parameter can potentially be lowered after each DSC iteration to increase the mesh complexity during the deformation.

6.3 DSC based topology optimization

DSC for topology optimization is using an explicit surface representation which means that there is no need for the material density as in the density approach described in section 3.3. DSC based topology optimization as described in [CBS] uses both nodal movement and element relabeling for finding good solutions. An overview of the method is illustrated in figure 6.4 which for simplicity only shows the shape and not the full design domain discretization. First, the global stiffness matrix $\mathbf{K}$ is assembled and solved to compute the nodal displacement vector $\mathbf{u}$. Finally the optimized shaped is found using element relabeling and/or nodal movement based on the sensitivity analysis as described below. These steps are repeated until convergence.

In order to avoid local optima and minimize a jagged surface, the preferred FE element type is Linear Strain Tetrahedron (LST) with 10 d.o.f. per element instead of the Constant Strain Tetrahedron (CST) with 4 d.o.f. [CNJA+14].

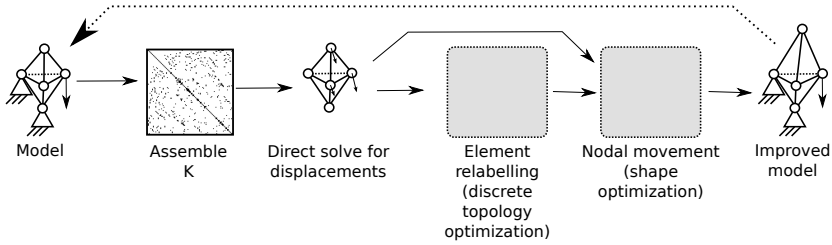


Figure 6.4: Overview of DSC based topology optimization.

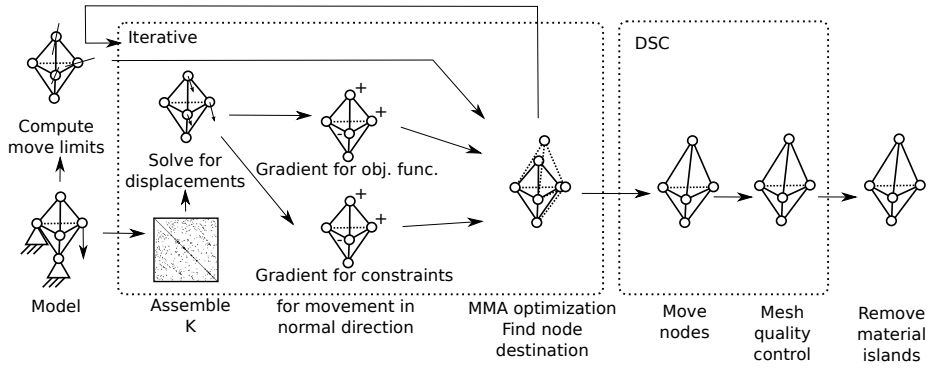


Figure 6.5: Optimizing shape by moving nodes along the normal direction.

6.3.1 Nodal movement

The interface of the object is moved to improve the objective function, such as compliance, while satisfying the constraints as illustrated in figure 6.5. In order to reduce the d.o.f. each node is restricted to move only in its normal direction of the surface. This simplification is based on the observation that motion in the tangent direction of the surface will not change the shape much.

The design variables in the optimization is the displacements $\mathbf{x}$ in the normal direction $\mathbf{n}$ where the goal is to find the new position $p(\mathbf{x})$ based on the original position $\mathbf{x}^0$:

$$p_i(x_i) = x_i^0 + x_i \cdot n_i \tag{6.1}$$

where i is interface nodes. The optimization problem can then be posed as:

$$\begin{aligned}
\min_{\mathbf{x}\in R^n} \quad &: \quad c(\mathbf{x}) && \text{(Objective function)} \\
\text{subject to} \quad &: \quad g_i(\mathbf{x}) \leq 0 \quad , i = 1, \ldots, q && \text{(Constraints)} \\
&\quad\quad \mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f}(\mathbf{x}) && \text{(FEM equilibrium)} \\
&\quad\quad \mathbf{x}_{min} \leq \mathbf{x}_e \leq \mathbf{x}_{max} \quad e = 1, \ldots, n && \text{(Design variables)}
\end{aligned} \tag{6.2}$$

where $g_i(\mathbf{x})$ is the constraints (e.g. volume constraint), $\mathbf{f}(\mathbf{x})$ is the region based load, $c(\mathbf{x})$ is the objective function, such as compliance. The $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are move limits based on surrounding simplicies, such that movement will not cause any simplex to become degenerate or flipped.

This is a smooth, non-linear optimization problem and can be solved using a gradient based solver, such as the Method of Moving Asymptotes (MMA) [Sva87].

In order to be able to compute the gradient, we need the global stiffness matrix $\mathbf{K}$ to be non singular at any given time. To ensure this the optimization is initiated with a full volume fraction, which are then lowered to the specified volume fraction over a number of iterations. An alternative way to ensuring non singularity is to start with an arbitrary figure, which connects all loads and supports. One of the performance tricks used is that the FEM is computed using only elements with material.

6.3.2 Element relabeling

Simplicial complexies are relabelled to speed up the optimization as well as to introduce holes inside the structure. Creating internal holes is important since this will prevent some local optima which can occur when only node movement is used. The relabeling is based on the gradient of the objective function and the constraints as shown in figure 6.6.

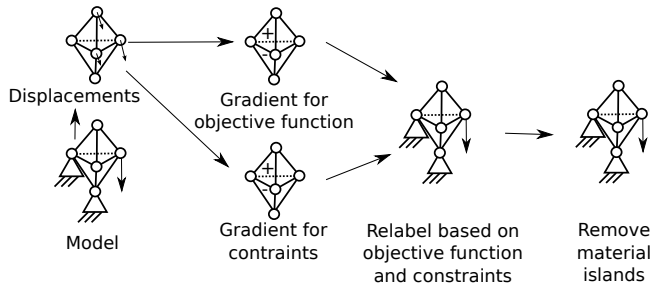


Figure 6.6: Relabeling used in DSC based topology optimization.

The relabeling can be formulated as a discrete optimization problem:

$$\begin{aligned} \min_{\mathbf{m}\in R^n} \quad &: \quad c(\mathbf{m}) && \text{(Objective function)} \\ \text{subject to} \quad &: \quad g_i(\mathbf{m}) \leq 0 \quad , i = 1, \ldots, q && \text{(Constraints)} \\ &\quad \mathbf{K}(\mathbf{m})\mathbf{u} = \mathbf{f}(\mathbf{m}) && \text{(FEM equilibrium)} \\ &\quad m_e \in \{m_1, \ldots, m_n\} && \text{(Design variables)} \end{aligned} \tag{6.3}$$

where n is the number of simplex elements. Since the relabeling is combined with the continuous optimization using node movements it does not have to be solved optimally. Instead of solving the computationally hard optimization problem, an approximate solution is found based on heuristic strategies as described in [CBS]. These heuristic strategies are based on the theory of topological derivatives ([EKS94], [SZ99], [GGM00],[FNTP03], [DGAJ08]). The topological derivative is the change in objective function c when introducing an infinitesimal hole in element e.

6.4 Topology optimization based modeling tool

There exists many types of tools for creating and modeling virtual 3D shapes. One such tool is digital sculpting tools, where the user modifies the shape of an object using drag-based strokes using a mouse or a digitizing tablet. Each stroke correspond to a local operation on the shape, such as adding material,

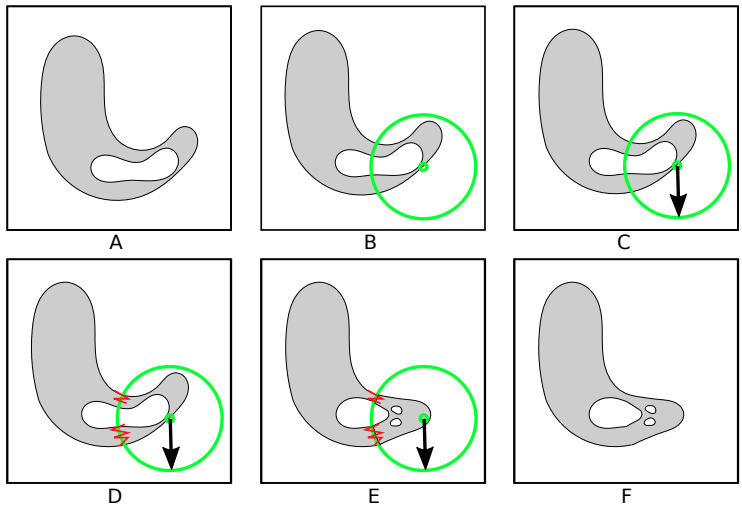


Figure 6.7: Modeling using DSC based topology optimization. (A) Initial shape (B) Selection of subdomain (C) Adding point load (D) Subdomain is fixed where material is located at the boundary (E) Subregion will be topology optimized for minimum compliance where volume is maintained (F) Final shape

removing material, smoothing, sharpening or pinching. Sculpting tools are primarily used for designing organic shapes, such as humans and animals. The user interaction design used in sculpting tools is heavily inspired by clay modeling and hence the terminology used often refers to this. Sculpting tools stand out from other 3D modeling tools by providing a very direct way of modeling with purely local control and high predictability. There exists many commercial and open source programs which provides digital sculpting capabilities, such as ZBrush and Blender.

The idea of a topology optimization based modeling tool is heavily inspired by digital sculpting tools. On an existing shape the user can optimize the material distribution locally in a specified subdomain using the tool illustrated in figure 6.7. The size of this subdomain can be changed by the user, and hereby changing the size of the local operation. A shape change is initiated by a simple gesture, where a click defines the position of a load (6.7-B), followed by a drag gesture to specify the direction of the load (6.7-C). The shape is supported at the boundary of the region of interest (6.7-D). The tool will now optimize the subdomain while maintaining the volume of the shape (6.7-E). The topology optimizer will continuously optimize the shape as long as the drag gesture is active (6.7-E).

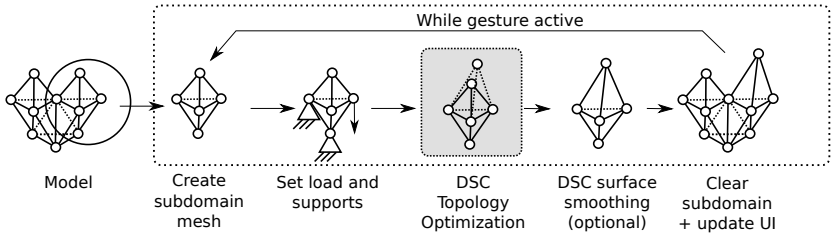


Figure 6.8: Overview of DSC topology optimization based modeling.

The goal of the tool is to use topology optimization as a tool to find interesting and aesthetically pleasing shapes, rather than searching for global solutions to physically based optimization problem.

6.5 Implementation

One of the key challenges is to make the tool interactive and provide visual feedback within reasonable time. To speedup the topology optimization computations the subdomain support and parallelization has been added to DSC. Also raytracing capabilities have been added to improve the responsiveness of the user interface.

Figure 6.8 shows how the subdomain is used to only run the topology optimization on elements within a subdomain in order to achieve an interactive update rate. An optional smoothing step can be performed on movable nodes in the subdomain to reduce the occurrence jagged edges.

6.5.1 DSC subdomain support

One essential problem is that DSC work on the full domain, where as this tool only make small local changes and hence only local mesh quality control is required.

To solve this problem DSC has been modified to supports subdomain, which essentially flags elements outside the subdomain as excluded. The iterators for the simplex types (tetrahedrons, faces, edges and nodes) will not include any excluded element and hence quality control and other algorithms using DSC will only operate on the subset. From a performance perspective the subdomain

feature is nearly free, since the simplex types already have a status flag, which is checked during iteration to only include active elements. The only significant performance cost is setting a subdomain, which involves identifying which tetrahedrons are inside the subdomain and finding their associated faces, edges and nodes. One important detail is that excluded elements are still accessible when traversing the data structure from a single element (using the boundary and co-boundary relations of a simplex). This allows algorithms which use information from the neighborhood of an element to still execute correctly - smart Laplacian smoothing is an example of such algorithm.

6.5.2 Parallelization of DSC

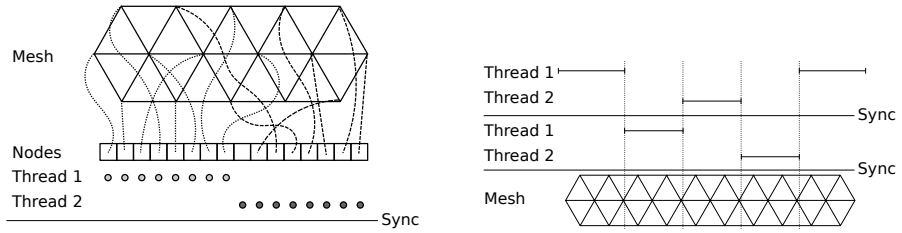
While the DSC based topology optimization library uses multiple threads to speed up the computation time, the DSC itself was not designed for multi-threading. For this reason the DSC has been redesigned to natively support multiple treads. The parallelization extension is created using two C++ template functions which works with all of the four simplex types. The number of parallel worker threads used can be adjusted if other than the number of available hardware threads is needed.

for_each_par()

This extension adds parallelization to DSC using the fork-join execution model. The `for_each_par()` function takes a single lambda function as parameter which is executed on all valid elements on the given simplex type. The execution is performed by dividing the internal simplex storage among the number of threads used. Then all threads are launched, each of which is executing the lambda expression on its subset. This approach has the benefit that each thread access its elements in a memory coherent way (since the elements are located sequentially in memory). After all worker threads have been launched the main threads will wait for all worker threads to finish before continuing execution. A conceptual illustration is found in figure 6.9-a. In case the DSC has requested only one thread, then everything is executed in the main thread.

One limitation of the extension is that the execution is not synchronized in any way. This means that it cannot be used for updating elements based on values read from other elements, due to potential race conditions. This limitation also includes creation, deletion and changing relation of simplices.

The extension is now used inside DSC to speed up basic operations, such as `find()` and `get_max_edge_length()`.



(a) `for_each_par()` **(b)** `for_each_par_sp()`

Figure 6.9: Parallelization of DSC - here shown for nodes. (a) shows how each thread process a subset of the kernels list of nodes. (b) show how the thread iterations is based on the spatial position of the nodes.

for_each_par_sp()

To support cases where updating elements are based on data from neighbor elements, a space partitioning version of the `for_each_par()` is added. The idea here is to partition the domain in one dimension, such that there is at least two tetrahedra between two neighbor partitions. The parallelization is then first evaluated on every second partition, where each partition is assigned to a thread. Afterwards, the remaining partitions are evaluated. A conceptual illustration is shown in figure 6.9-b.

This evaluation scheme allows reading from neighbor simplicies when updating values relating to a simplex, without the risk of data races. The total runtime is slower than `for_each_par()` due to time spend on partitioning and due to a less memory coherent data access.

The extension is now used for in DSC to speed up smoothing.

6.5.3 DSC raytracing

When using DSC in an interactive manner, it is often important to find the face of the interface based on a location in screen-space, for instance such that a face can be selected based on a mouse click. To solve this problem in an efficient manner, a raytracer for the simplex data structure of DSC has been implemented. The raytracer works efficiently by finding entry and exit points for a tetrahedron and then use the simplex relations to find the adjacent tetrahedron. Using the spatial relationship to search the DSC data structure is a performance improvement compared to naively testing all faces of the mesh for intersections. The concept is illustrated in figure 6.10.

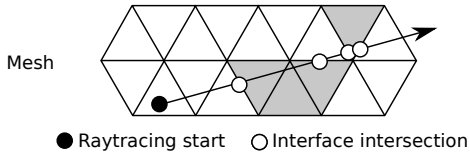
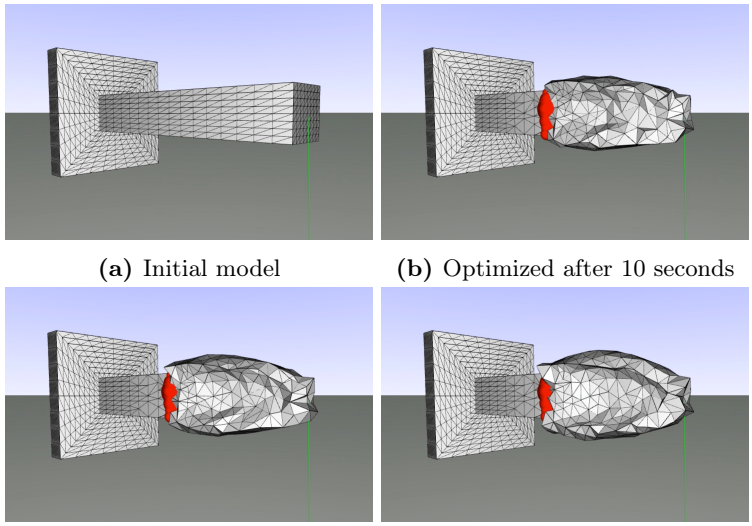


Figure 6.10: Raytracing DSC here returning only interface faces.



(a) Initial model **(b)** Optimized after 10 seconds

(c) Optimized after 15 seconds **(d)** Optimized after 25 seconds

Figure 6.11: Example of DSC based topology optimization modeling. The red area is supported elements. The tiny green line is load.

6.6 Results

We have tested the tool on a standard example. The example adds a vertical load to a simple cantilever is shown in figure 6.11. The subdomain radius is limited to only affecting half of the beam, which is visualized using the red support region. The result has the expected I-profile, which can also be found using the TopOpt 3D app as shown in figure 6.12.

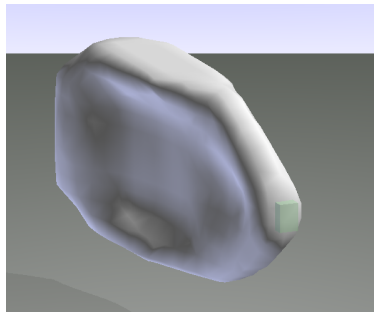


Figure 6.12: Cantilever example created using the TopOpt 3D app

6.7 Discussion

One limitation of the proposed tool is performance. While the tool provides a reasonable framerate when dealing with small problems such as figure 6.11, the update rate drops significant when used for larger regions. This problem can be improved by either using faster hardware with more and/or faster threads or improving the parallelism of the implementation.

Since the structure is initiated with an active volume constraint, the discrete topology optimization step does usually not remove any elements. Instead the optimization is mainly based on the continuous shape optimization. A related problem is that since the optimization only moves the interface based on the interface node gradients, there is no way for the structure to 'sense' other options for support. Both issues may lead some local optimum. A solution could be to fill the subdomain with material and slowly lower the volume fraction to the initial value which is also the approach used in [CBS]. This approach will however challenge the interactive nature of the tool, since this requires many more iterations and the initial iterations will be more computationally heavy.

Alternatively, the topology optimizing modeling tool could also be implemented using the density approach, which also has been used in the TopOpt apps described in the previous chapters. This would also cure the above mentioned problems, but using the density approach would require that the subdomain is translated back and forth between explicit and implicit surface representations during optimization. It also involves 'stitching' the result back to the full structure. These problems are avoided completely when using an adaptive mesh such as DSC, where any surface details down to a specified threshold are maintained and the same representation can used for visualization, optimization and manufacturing.

Chapter 7

Discussion and conclusion

The overall goal of this thesis has been to explore how to use topology optimization to create new and innovative applications running in an interactive manner. The thesis has a broad focus covering both the human-computer interaction, used for creating well-designed user interfaces, computer science, used for parallelism of the code and cross-platform development, and mechanical engineering, for structural analysis and topology optimization.

Interactive topology optimization is now possible both on PCs and handheld devices primarily due to the increase in computational power as well as the highly optimized libraries and methods available. When designing such new types of applications it is often needed to rethink the user interface and user interaction in order fully exploit the full potential of the interactivity. The main reason for this is that such applications have fundamental differences from existing applications. Instead, a good user interface design can be created using the user interface design principles and the usability can be evaluated using the usability evaluation methods both described in chapter 2.

TopOpt App and TopOpt 3D app are both first applications of their kind for interactive topology optimization in 2D and 3D. The applications show a new way of modeling shapes using an indirect approach where user specifies the constraints the shape must obey while the shape emerges through topology optimization. The high performance of both apps is achieved using the MTOP

approach for the 2D case and the MG-PCG method for the 3D case. Another important performance gain comes creating a native C++ optimization kernel using our library OOCholmod as an elegant and efficient abstraction of the low-level libraries SuiteSparse / Cholmod, BLAS and LAPACK. In addition the FFEM app shows how the code and user interface of the TopOpt App can easily be adapted to solve a similar problem.

The TopOpt Game shows how topology optimization can be gamified, by asking the player to create a shape in order to optimize his score (and minimize compliance). During gameplay the player can constantly see the consequences of his actions in the score-graph, which can also be used for backtracking to an earlier state. By analysing the game data we have found that playing the game will improve the players intuition of topology optimization.

The presented applications have all been designed to work well on both PCs and handheld devices with touch-screens. The user interface design is therefor constrained both in terms of how much information can be displayed on the screen as well as the accuracy on the input. Throughout the development of the applications, their usability has been evaluated primarily using heuristic evaluation and informal user based evaluations, where people are asked to try the application while their interactions are being observed.

One way to measure the success of the applications is to look at download statistics and user reviews in Appendix E. The TopOpt App is the most popular app, which is probably due to that is has been available for a long time and that introductions to topology optimization often focuses on 2D problems. Users have also provided very positive feedback for both the TopOpt App and the the FFEM App where the average ratings are between 4.0 and 4.8 out of 5. The TopOpt Game has relative low download numbers. We believe the reason for this is that people are unaware of its existence and hope to see a significant increase once Paper C gets published. Overall, we are very pleased with the download numbers as well as the positive feedback and see this as a verification of well-designed user interfaces and high performance code.

Modeling using DSC Based topology optimization shows how topology optimization can be used in a completely new way as a local modeling tool with a very direct user interaction. The method uses the DSC for storing and deforming the structure which has the benefit of having an explicitly defined interface. This avoids loosing details which would likely occur if the structure had to be transformed to an implicit representation. To ensure interactiveness the DSC has been extended with support for parallelism as well as subdomains. The current result is an interactive proof of concept. It has not been made public available since the application is still a bit too sensitive to quality parameters used in both DSC as well as DSC based topology optimization.

This thesis has shown how topology optimization can be used interactively in applications for both 2D and 3D. Interesting future work includes optimization of coated structures as described in [CAS15] where the optimized structure has a coating with a prescribed thickness and different material properties. It could also be interesting to explore how the many sensors on the handheld devices could be used. Currently the accelerometer and gyroscope are used for the gravity loads in the TopOpt 2D, but could also be used for controlling both the view orientation as well as gravity loads in the 3D app. The upcoming Force Touch sensor for Apple devices could also be used to vary load magnitude based on the pressure level. The camera could be used to capture design of existing structures, such as bridges, for a structural analysis in the FFEM app.

From informal empirical studies we know that people generally find the user interface in apps simple and intuitive to use. For the 3D app the ViewCube provides an easy way to control the view, however this only works for simple and small structures. A new view interaction scheme would be needed if the app is extended to more complex structures. The same is also true for the 2D app, where zoom and pinch gestures would be needed if the resolution is increased significant.

Hopefully, the work of this thesis will inspire new, innovative and interactive applications both within topology optimization and in science in general.

Appendix A

Paper A: Interactive topology optimization on hand-held devices

Aage, N., Nobel-Jørgensen, M., Andreasen, C. S., Sigmund, O. (2013). Interactive topology optimization on hand-held devices. *Structural and Multidisciplinary Optimization* 47.1: 1-6.

Struct Multidisc Optim (2013) 47:1–6
DOI 10.1007/s00158-012-0827-z

EDUCATIONAL ARTICLE

Interactive topology optimization on hand-held devices

Niels Aage · Morten Nobel-Jørgensen · Casper Schousboe Andreasen · Ole Sigmund

Received: 29 March 2012 / Revised: 15 June 2012 / Accepted: 24 June 2012 / Published online: 19 July 2012

Abstract This paper presents an interactive topology optimization application designed for hand-held devices running iOS or Android. The TopOpt app solves the 2D minimum compliance problem with interactive control of load and support positions as well as volume fraction. Thus, it is possible to change the problem settings on the fly and watch the design evolve to a new optimum in real time. The use of an interactive app makes it extremely simple to learn and understand the influence of load-directions, support conditions and volume fraction. The topology optimization kernel is written in C# and the graphical user interface is developed using the game engine Unity3D. The underlying code is inspired by the publicly available 88 and 99 line Matlab codes for topology optimization but does not utilize any low-level linear algebra routines such as BLAS or LAPACK. The TopOpt App can be downloaded on iOS devices from the Apple App Store, at Google Play for the Android platform, and a web-version can be run from www.topopt.dtu.dk.

Keywords Interactiveness · Topology optimization · Smartphones · Tablets · PDE constrained optimization

The authors acknowledge the support from the Villum foundation through the NextTop project.

N. Aage ( ) · C. S. Andreasen · O. Sigmund
Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Nils Koppels Alle, B.404, 2800 Kgs. Lyngby, Denmark
e-mail: naa@mek.dtu.dk

M. Nobel-Jørgensen
Department of Informatics and Mathematical Modelling, Technical University of Denmark, Asmussens Alle, B.305, 2800 Kgs. Lyngby, Denmark

1 Introduction

In Denmark more than 50 % of all households are in possession of a smartphone (Statistics Denmark 2012), and it is expected that the percentage is much higher amongst households with engineering, design and architecture students. Not only do smartphones provide the user with easy access to gadgets such as GPS, gyroscopes, accelerometers, etc., they also contain powerful processing units that can be used for advanced scientific computing and high-level educational tools as demonstrated in this paper.

The topology optimization method is a numerical and iterative structural optimization tool which optimizes the material distribution in a specified design domain in order to maximize stiffness or other objectives, typically subject to a volume constraint. Since 1999, the TopOpt-group has been hosting a web-based topology optimization Applet (www.topopt.dtu.dk, "Applets and Software", "Server side applets", "Compliance Design") (Tcherniak and Sigmund 2001). The Applet has been extensively used by engineering, architectural and industrial design students and practitioners as well as a general audience and has by now (March 2012) been run over 210,000 times by more than 13,000 unique users. This code is passive in the sense that the user selects design specifications (design domain, boundary conditions, volume fraction, etc.), presses a submit button and then waits for the optimization to finish (in usually 5–10 s) before he/she sees an animation of the design process on his screen. The code is run on a server and hence the response time depends on the number of active users at any given time and on the speed of the internet connection.

In this paper we present a fully interactive topology optimization application, the TopOpt App, which solves the 2D minimum compliance problem with interactive control of loads and supports as well as the amount of available

Fig. 1 Picture of the TopOpt App being used on an iPad

material. The App is developed partially to provide a more interactive user interface than the existing web applet and partially to port the web applet to mobile platforms. Based on an efficient code that is executed directly on the device, and not server-side, the users will have the impression that the structure becomes "alive"—constantly adapting to varying loads and boundary conditions. With this set-up it is extremely simple to learn and understand the influence of load-directions and support conditions and to develop a general understanding and intuition for structural design by the topology optimization method. A picture of the TopOpt App running on an iPad is shown in Fig. 1.

The paper is organized as follows. The topology optimization problem is presented in Section 2 along with motivation for the chosen approach. Section 3 presents the framework used to develop the TopOpt App with the interactive graphical user-interface (GUI) and the C# optimization kernel. Section 4 presents snapshots of the applet as well as a discussion of the different issues that arise when allowing the optimization problem to be modified on the fly. Section 5 summarizes our findings and gives some directions for future extensions and applications.

2 Problem formulation

The minimum compliance problem is a classical topology optimization problem in which the goal is to maximize the stiffness of a structure subject to a constraint on the available material (Bendsøe and Sigmund 2004). The requirements to the implementation presented in this paper differ slightly from standard implementations, including the ones found in Sigmund (2001) and Andreassen et al. (2010), which otherwise form a basis for the optimizer presented here. The TopOpt App does not only require a fast solver to maintain a high frame rate. More importantly, the solver must be capable of starting from any given 0–1 design, and from this, evolve to a new optimum. Furthermore, the way in which the design evolves when changing loads and supports, must also look and feel right to ensure that the user experiences the structure as being alive and constantly adapting to changing conditions. Although the demands on the optimization solver are somewhat different, the minimum compliance problem remains unchanged and can be stated in a discrete form as

$$\begin{aligned} \min_{\rho\in\mathbb{R}^n} \quad & \phi(\boldsymbol{u}(\boldsymbol{\rho}),\boldsymbol{\rho}) = \boldsymbol{F}^T\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{F} \\ & V(\boldsymbol{\rho})/V^* - 1 \le 0 \\ & 0 < \rho^{\min} \le \rho_i \le 1, \quad i = 1,n \end{aligned} \tag{1}$$

where $\phi = \boldsymbol{F}^T\boldsymbol{u}$ is the compliance, $\boldsymbol{\rho}$ is a vector of n densities (design variables), $\boldsymbol{u}$ and $\boldsymbol{F}$ are the nodal displacement and force vectors, respectively, $\boldsymbol{K}(\boldsymbol{\rho})$ is the global stiffness matrix , and $V(\boldsymbol{\rho})/V^* - 1 \le 0$ is the volume constraint. Finally, $\rho^{\min}$ is a lower bound on the design variables. The elasticity equations are solved by the finite element method using four node bilinear elements (see e.g. Zienkiewicz and Taylor 2000).

2.1 Design representation

The design representation for the TopOpt App has to provide the user with a smooth interactive experience, and the design representation should be as fine as possible. To accommodate the smoothness it was by numerous numerical experiments found that the SIMP scheme (Bendsøe 1989; Zhou and Rozvany 1991; Mlejnek 1992) with a high lower bound on both density and stiffness gives the best results. The implemented stiffness interpolation is given as

$$E(\rho_i) = 0.01 + 0.99\rho_i^p \tag{2}$$

where $p = 3$ is the penalization factor. The lower bound on stiffness $E_{\min} = 0.01$ is chosen relatively high, since this has shown to yield the best performance when the design has to evolve from one already determined optimum to the next. We have found that a higher value ensures better behavior when a load or a support suddenly is moved to a void region, however, a higher value also adds a risk of introducing artificial stiffness of void regions that may alter the design in undesired manners. The same argument applies to the choice of lower bound for the densities, which is set to $\rho^{\min} = 0.01$. During the development of the TopOpt App the RAMP interpolation scheme (Stolpe and Svanberg 2001) was tested as an alternative to SIMP. However, at the end, the SIMP scheme was selected for its superior stability.

Though the CPUs of smartphones have evolved tremendously during the past few years, they are still lacking behind the performance of laptop, or desktop, CPUs. The speed ratio for the TopOpt App running on the IPhone 4S, Ipad 2 or a high level laptop is approximately 1:1.2:12. Therefore we have employed the multiresolution (MTOP) design representation from Nguyen et al. (2010), to obtain a finer design representation with little extra computational cost. The MTOP approach divides every finite element into several design elements. For the TopOpt App we use four design variables for every finite element as illustrated in Fig. 2. Thus, for each finite element the element matrix contribution can be given as

$$\boldsymbol{K}_e(\rho_e) = \sum_{i=1}^{4} \boldsymbol{K}_0^i \rho_e^i \tag{3}$$

where $\boldsymbol{K}_0^i$ is the reference stiffness matrix evaluated at each of the four design variable locations. Hence, the MTOP approach can be compared to a Gaussian integration with a piecewise constant density variable for each integration point. The major benefit of the MTOP approach is that although the assembly becomes four times more expensive, the size of the linear equation system to be solved remains the same. Since the mesh used is regular, the numerical integration of the four sub-matrices can be done once and reused for all finite elements throughout the iteration process. Although the MTOP approach yields a finer design representation, it is important to note that the checkerboard problem depends on the standard finite element discretization and hence any subsequent filtering must be performed with a radius comparable to the physical element size.

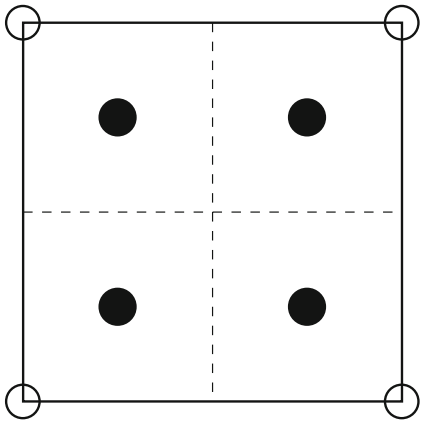


Fig. 2 Illustration of the multiresolution (MTOP) approach used for the TopOpt App. The *full lines* represent the displacement element and the *circles* indicate displacement nodes. The *dashes lines* show the distribution of design variables, ρ_i, within each physical element and the *filled circles* their location

To alleviate checkerboarding we apply the sensitivity filter (Sigmund 1997). The filter operator can be stated as seen below following the matrix approach from Andreassen et al. (2010)

$$\frac{\partial \hat{\phi}}{\partial \rho_i} = \frac{1}{\rho_i \sum\limits_{j \in N_i} H_{ij}} \sum_{j \in N_i} H_{ij} \rho_j \frac{\partial \phi}{\partial \rho_j} \tag{4}$$

where N_i is the index set of design variables within a radius $r_{\min}$ from design variable ρ_i, and the weight factor H_{ij} is given by the sparse matrix

$$H_{ij} = \max(0, r_{\min} - \mathrm{dist}(\rho_i, \rho_j)) \tag{5}$$

where $\mathrm{dist}(\rho_i, \rho_j)$ is the distance from design variable ρ_i to variable ρ_j. The filter radius is set to $r_{\min} = 2.6$ times the design element size (i.e. 1.3 times the finite element size). Note that we use the sensitivity filter since, again by extensive numerical experiments, it tends to provide smoother convergence and is better to avoid getting stuck in local minima compared to the density filter (Bruns and Tortorelli 2001; Bourdin 2001).

The design update is performed using the optimality criteria approach as implemented in Andreassen et al. (2010).

Before the optimized design is displayed on screen it is post-processed in the following way. It is first projected onto a two times refined design mesh and subsequently filtered using the standard density filter using a filter radius equivalent to two element sizes on the refined mesh. Finally, to make the interface between void and material sharper, the refined and filtered design is projected using a Heaviside step function as presented in Wang et al. (2011), i.e.

$$\tilde{\rho}_i = \frac{\tanh(\beta\eta) + \tanh(\beta(\rho_i - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \tag{6}$$

with a sharpness control of $\beta = 6$ and cut-off point of $\eta = 0.5$. Note that the final steps of filtering and projection are performed outside the optimization framework, and thus simply act as an image processing technique to enhance the design resolution. Basically, the post-processing step corresponds to a smoothed thresholding of the density field.

3 Implementation

The TopOpt App has been developed in the Unity3D game engine (Unity Technologies 2012), which was chosen due to its cross-platform portability and due to the presence of in-house expertise. The multi-platform support means that the same optimization kernel can be used for Android,

iOS and web releases with only minor modifications to the layout of the GUI and user-interaction. However, the cross-platform support capability comes at the cost of generality which hinders the use of optimized linear algebra libraries such as BLAS and LAPACK which are only available for certain platforms. Therefore our optimization kernel is built from scratch including sparse matrix classes containing linear solvers, assembly operators, matrix-vector multiplication, etc.

3.1 Optimization kernel

The TopOpt kernel contains an optimization solver equivalent to that presented in the 99 line Matlab code paper (Sigmund 2001), but is written entirely in C# since this is the language supported by Unity3D. Due to the object oriented nature of C# it is straight forward to implement sparse matrix classes which can be used for both filtering and linear solver. Using sparse matrices for filtering is described in detail in Andreassen et al. (2010), and the major advantage of this approach is that the neighborhood search only has to be done once while assembling the filter matrix. The filter can then be applied as a single sparse matrix-vector product followed by a scaling operation. The sparse matrix approach to filtering is generally faster than performing a quadruple for-loop for each filtering operation as done in the 99 line Matlab code paper (Sigmund 2001). This is mainly because the for-loop approach requires the evaluation of (4) and subsequently (5) numerous times at each iteration. Especially (5) is expensive since it both involves a max-statement and a square root. Note however, that if it was not for the benefit of the simple way to perform filtering using sparse matrices, a banded solver could just as well have been used since our tests have shown that a banded solver matches the performance of the implemented sparse solver.

During the development phase we have also experimented with iterative solvers and the multigrid preconditioned conjugate gradient method in particular. Although this solver can outperform the direct solver when utilizing a smart stopping criteria as described in Arioli (2004) and tricks from re-analysis (Amir and Sigmund 2011), it yields undesirable iterations when changing the optimization settings. That is, due to an inexact FE solution the sensitivities in parts of the domain may be wrong, which results in material appearing and disappearing at seemingly random locations in brief glimpses. Since the App is intended to provide a natural transition between optima, the slower, yet more stable, direct solver is used.

Due to the interactive nature of the App, it is very easy to pose a problem which is infeasible or numerically ill-posed. This could for example be a result of missing loads, inadequate supports (singular system), possible free modes (close to singular system) or any combinations of such. If any of the above problems are detected, the optimization solver is frozen and an error message is displayed until the user resolves the problem.

As a final remark our experience has shown that for very coarse discretizations superior stability is achieved for a larger filter radius, e.g. $r_{\min} = 1.4$.

3.2 GUI

The GUI can be seen in Fig. 3 and consists of the following items—all conveniently implemented using the Unity3D game engine which, as mentioned in previous sections, facilitates an easy way to port the code between different platforms. The move symbol allows the user to freely move both loads and supports between nodes in the underlying finite element mesh. The rotation symbol provide the possibility to rotate forces and supports. Note that rotating the supports only has a physical meaning for the simple supports. The scaling operator, illustrated by a square frame with an arrow attached to the top left corner, is used to distribute forces and supports. The downward arrow denotes a nodal force with unit magnitude, independent of load orientation. Note that the unit magnitude is maintained also for a distributed load. Furthermore, it should be mentioned that a single nodal force combined with a distributed load, in some cases, can result in designs for which the distributed load is not fully supported. This issue will be addressed and fixed in an upcoming version of the App. The two supports represent simple and fixed nodal supports, respectively. The trash can is used to delete items from the optimization domain, and the white triangle is used to change the volume fraction. All the above mentioned GUI objects are fully interactive,

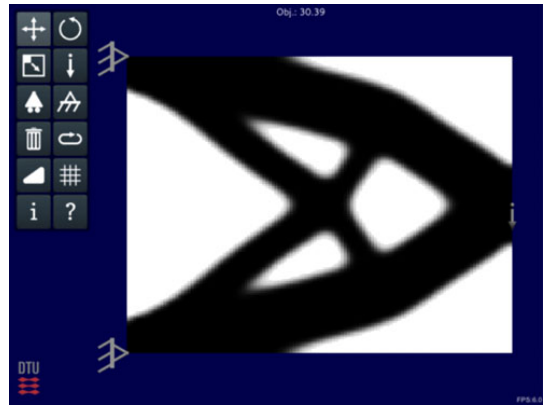


Fig. 3 Screenshot of the interactive TopOpt App. The menu can be seen to the *left* and the optimization domain to the *right*. Centered at the top the current value of compliance is shown. In case of un-physical settings an appropriate error message is shown instead. At the bottom right corner the frame-rate is shown

meaning that any modification will change the optimization problem from the next frame (iteration) and on.

The remaining symbols are comprised of the following functionality: A restart feature denoted by an elongated circle, is used to restart the optimization solver with a uniform material distribution equal to that of the current volume fraction. This is needed since the optimizer can, and does, reach local optima, which can be alleviated by a restart. The grid button allows the user to change between a number of fixed mesh resolutions and will restart the entire optimization code when a new grid size is selected. The final two items in the menu yield a short summary of the underlying optimization problem and solution approach and a help menu, denoted with **i** and **?**, respectively.

4 Discussion

The TopOpt App is mainly intended for educational purposes, and not as a commercial optimization tool. It can, however, readily be used as inspiration in the early stages of a design process in e.g. architecture, industrial design and engineering. Figure 4 shows a series of screenshots from running the App, demonstrating how the design evolves from one optimum to another. As for most topology optimization problems, the major changes in topology take place within the first 30 design cycles. This means that although the TopOpt App should solve the optimization problem as fast as possible, more than 30 frames per second will hinder the user in following the design evolution. In order to get a reasonable speed we therefore adapt the discretization to the device. For example on an iPhone 4S the standard density mesh is chosen as 88×64 (i.e. 22×16 4-node finite elements), which yields a satisfactory eight frames per second. For slower or faster devices, the user may select coarser or finer discretizations.

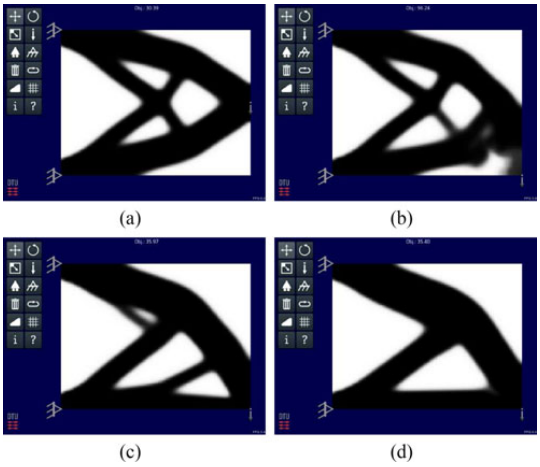


Fig. 4 The screenshots in (**a**) through (**d**) show how the TopOpt app evolves from one optimum to another

5 Conclusions

This paper presents a new interactive topology optimization applet, the TopOpt App, for minimum compliance problems. The App can be run on smartphones with iOS and Android, and as a web application. The TopOpt App both demonstrates the capabilities of smartphones in terms of CPU power, but also a new way to perform topology optimization by real-time interaction. The objective of the App is to provide engineering, design and architect students and practitioners with a fast and simple way to use topology optimization. This may lead to a better understanding of optimal material distributions with respect to changes in load conditions, support conditions and the amount of available material. Apart from the educative possibilities, the App may also be interesting to play with for seasoned topology optimization practitioners. For example, it is mind-boggling to see how much a design can change by simply moving a point load or support from the domain corner and one element length into the domain.

The TopOpt App is the first mobile App to offer topology optimization. Future versions will include multiple loads and passive domains as offered by its passive predecessor still found at the www.topopt.dtu.dk web-site.

With the publication of this App, we hope to inspire the topology optimization community as well as the mechanics community in general to provide educative Apps that can be used to help students in understanding complex mechanical topics. One may just think of the smartphones' built-in accelerometers and gyroscopes which, with the right App, could provide entirely new and inventive ways of teaching engineering dynamics.

Acknowledgments The authors would like to extend their gratitude to the members of the TopOpt and NextTop groups at DTU for their invaluable input on the design and testing of the TopOpt app.

References

Amir O, Sigmund O (2011) On reducing computational effort in topology optimization: how far can we go? Struct Multidisc Optim 44:25–29

Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2010) Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidisc Optim 43(1):1–16

Arioli M (2004) A stopping criterion for the conjugate gradient algorithm in a finite element method framework. Numer Math 97:1–24

Bendsøe M (1989) Optimal shape design as a material distribution problem. Struct Optim 1:193–202

Bendsøe M, Sigmund O (2004) Topology optimization; theory, methods and applications, 2nd edn. Springer, Berlin Heidelberg New York

Bourdin B (2001) Filters in topology optimization Int J Numer Methods Eng 50(9):2143–2158

Bruns TE, Tortorelli DA (2001) Topology optimization of non-linear elastic structures and compliant mechanisms. Comput Methods Appl Mech Eng 190(26–27):3443–3459

Mlejnek HP (1992) Some aspects of the genesis of structures. Struct Optim 5:64–69

Nguyen TH, Paulino GH, Song J, Le CH (2010) A computational paradigm for multiresolution topology optimization (mtop). Struct Multidisc Optim 41:525–539

Sigmund O (1997) On the design of compliant mechanisms using topology optimization. Mechan Struct Mach 25(4):493–525

Sigmund O (2001) A 99 line topology optimization code written in MATLAB. Struct Multidisc Optim 21(2):120–127

Statistics Denmark (2012) http://www.dst.dk/statistik/nyt/emneopdelt.aspx?psi=1409. Accessed 14 Jun 2012

Stolpe M, Svanberg K (2001) An alternative interpolation scheme for minimum compliance topology optimization. Struct Multidisc Optim 22(2):116–124

Tcherniak D, Sigmund O (2001) A web-based topology optimization program. Struct Multidisc Optim 22(3):179–187

Unity Technologies (2012) Unity3d. www.unity3d.com. Accessed 20 Mar 2012

Wang F, Lazarov B, Sigmund O (2011) On projection methods, convergence and robust formulations in topology optimization. Struct Multidisc Optim 43(6):767–784

Zhou M, Rozvany GIN (1991) The COC algorithm, part II: topological, geometry and generalized shape optimization. Comput Methods Appl Mech Eng 89(1–3):309–336

Zienkiewicz OC, Taylor RL (2000) Finite element method, vol 1, 5th edn. Butterworth-Heinemann

Appendix B

Paper B: 3D interactive topology optimization on hand-held devices

Nobel-Jørgensen, M., Aage, N. Christiansen, A. N., Igarashi, T., Bærentzen, J. A., Sigmund, O. (2014). 3D interactive topology optimization on hand-held devices. *Structural and Multidisciplinary Optimization* 51.6: 1385-1391

Struct Multidisc Optim (2015) 51:1385–1391
DOI 10.1007/s00158-014-1214-8

EDUCATIONAL ARTICLE

3D interactive topology optimization on hand-held devices

Morten Nobel-Jørgensen · Niels Aage · Asger Nyman Christiansen · Takeo Igarashi · J. Andreas Bærentzen · Ole Sigmund

Received: 26 September 2014 / Revised: 26 November 2014 / Accepted: 28 November 2014 / Published online: 20 December 2014

Abstract This educational paper describes the implementation aspects, user interface design considerations and workflow potential of the recently published TopOpt 3D App. The app solves the standard minimum compliance problem in 3D and allows the user to change design settings interactively at any point in time during the optimization. Apart from its educational nature, the app may point towards future ways of performing industrial design. Instead of the usual geometrize, then model and optimize approach, the geometry now automatically adapts to the varying boundary and loading conditions. The app is freely available for iOS at Apple's App Store and at http://www.topopt.dtu.dk/TopOpt3D for Windows and OSX.

Keywords Interactive · Topology optimization · Smartphones · Tablets

M. Nobel-Jørgensen (✉) · A. Nyman Christiansen · J. A. Bærentzen
Department of Applied Mathematics and Computer Science, Technical University of Denmark, Asmussens Alle, B.305, 2800 Kgs. Lyngby, Denmark
e-mail: mono@dtu.dk

T. Igarashi
Department of Computer Science, The University of Tokyo, Science bldg. 7, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan

N. Aage · O. Sigmund
Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Nils Koppels Alle, B.404, 2800, Kgs. Lyngby, Denmark

1 Introduction

Handheld devices such as mobile phones and tablets are becoming increasingly powerful and even beat supercomputers from the 90s in terms of pure computation power (Rooney 2011). This increased performance allows developers to create applications which solve hard problems in an interactive manner - the same problems that years ago ran as batch jobs on supercomputers. The performance as well as the number of different sensors, such as touchscreen, microphone, camera, and accelerometer, open up new opportunities for new and innovative applications.

Topology optimization (Bendsøe and Kikuchi 1988) is a numerical and iterative method for determining optimal material distributions. The classical example from structural mechanics is to maximize the stiffness of an elastic structure subject to a constraint on the amount of available material, i.e. the minimum compliance problem. Despite its simplicity, the minimum compliance objective has become an important design tool in e.g. the automobile and aerospace industries (Bendsøe and Sigmund 2003). The minimum compliance problem is also the topic of a number of educational applications and code examples. The first of such applications was a web-based applet developed by the DTU TopOpt group (Tcherniak and Sigmund 2001). To date this applet has been run almost a quarter million times by close to 15.000 unique users. In 2012 the TopOpt App (Aage et al. 2013) was released, introducing the concept of interactive topology optimization. The TopOpt App solves the 2D minimum compliance problem with interactive control of loads and supports and can be used on both desktop computers and handheld devices (iOS and Android). Furthermore, the computations are performed locally on the device. Through a simple graphical user interface, the user defines the topology optimization problem, including loads,

Fig. 1 TopOpt 3D running on an iPad

supports, volume fraction, etc., while the optimized solution appears in real time. The application is highly interactive, which is a natural consequence of the update frequency, i.e. time for a single design cycle, which exceeds 30 cycles per second on most newer handheld devices. The high update frequency gives the user the impression that the structure comes "alive" and keeps the user's full attention as discussed in Miller (1968) and later in Nielsen (1994). The TopOpt App has become highly successful and had been installed on more than 11,500 handheld devices and used by more than 8,500 unique web users as of November 2014.

In this paper we present the next generation of fully interactive topology applications, namely the TopOpt 3D App. The app targets both desktop computers and iOS handheld devices such as iPad and iPhone. Equivalent to the 2D app, the user defines the problem using loads and supports, while the program shows the optimized shape on-the-fly. The topology optimization solver runs on the local device and is capable of achieving interactive update rates[1] even on handheld devices.[2] The user can navigate around the design domain to see and interact with the 3D model from different view directions. A picture of the application running on an iPad can be seen in Fig. 1.

In addition to TopOpt 3D's obvious application within teaching and education, it also presents a completely new way to think of geometry generation. In classical CAD/CAE, the industrial designer, or engineer, first designs the geometry, then models the problem and finally performs the optimization. The work presented in this paper suggests the opposite. That is, the designer first specifies the problem in terms of loads, supports, etc, and then follows the evolution of the geometry as it is being *created* by the optimizer. On top of this, the designer is allowed to interact directly with the geometry generation as it takes place and the process thus provides maximum design freedom with minimal waiting time.

The remainder of the paper is organized as follows: The topology optimization problem is described in Section 2 along with motivation for creating the application. Section 3 describes the implementation of the optimization kernel as well as the user interface and the visualization of the result. Section 4 presents different results created by the applications and how the application can be used as valuable tool in education and as a geometry-generating tool. Section 5 summarizes our findings and gives some directions for future extensions and applications.

2 Problem formulation

The optimization problem solved in TopOpt 3D is the standard minimum compliance design problem for linear elasticity (Bendsøe and Sigmund 2003). Following a finite element (FE) discretization, the classical topology optimization problem can be stated in a discrete form using the density approach (Sigmund and Maute 2013) as

$$\begin{aligned} \min_{\rho \in \mathbb{R}^n} \quad & \boldsymbol{\phi(u,\rho) = F^T u} \\ \text{s.t.} \quad & K(\rho)u = F \\ & V(\rho)/V^* - 1 \leq 0 \\ & 0 < \rho^{\min} \leq \rho_i \leq 1\,, i = 1, \ldots, n \end{aligned} \tag{1}$$

where $\phi = F^T u$ is compliance, ρ is a vector of n densities (design variables), u and F are the nodal displacement and force vectors respectively, $K(\rho)$ is the global stiffness matrix, and $V(\rho)/V^* - 1 \leq 0$ is the volume constraint. Finally, $\rho^{\min}$ is a lower bound on the design variables.

The solution approach for the design problem in (1) is dictated by the interactivity requirement. That is, the design cycles should be very fast and the transition from a given optima to a new one should be smooth, i.e. avoiding poor local minima. Thus, the solution strategy used in this work can be seen as a direct extension of the approach used for TopOpt App (Aage et al. 2013). Any noteworthy differences will be clearified in the upcomming sections.

3 Implementation

To make the app available on multiple platforms, we have used the game engine Unity as a platform abstraction layer, and this minimizes platform-specific code. The user interface code is written in C# and the optimization kernel is written in C++ to achieve maximum performance. Due to the huge increase in computational complexity compared

[1] We define interactive update rates to be less than 10 seconds, which is about the limit for keeping the user's attention (Nielsen 1993).

[2] The best experience will be achieved using newer devices such as iPhone 5s and iPad Air.

to the 2D app, we have chosen to switch from the in-house developed C# math kernel to CHOLMOD (Davis et al. 2014) built on (optimized) BLAS/LAPACK (BLAS Basic Linear Algebra Subprograms 2014; LAPACK - Linear Algebra PACKage 2014).[3] Due to the large variety of Android distributions and hardware, we have decided only to release TopOpt 3D for iOS, Windows and OSX, for which we can guarantee the performance of the app. At time of writing, the framerate on an iPad Air is 1.5 fps and 6-8 fps on a desktop computer.

3.1 Optimization kernel

The most performance-critical functions in the optimization kernel are solving the linear elasticity equation, performing the sensitivity filtering and completing the design update. The common denominator for these functions is that they depend on efficient sparse and dense matrix operations, i.e. matrix assemblies, matrix-vector multiplications, etc. For sensitivity filtering and the optimality criteria update, the approach presented in the 88 line Matlab (Sigmund 2001) code is readily adopted. However, the solution to the FEM system of equations is not feasible with direct methods due to the huge amount of time required for the matrix factorization. To alleviate this potential bottleneck, and in some sense make the 3D interactivity possible, we apply a geometric multigrid preconditioned conjugate gradient (MG-PCG) method for the solution of the linear system. The MG-PCG solver, as well as its numerical implementation, is described in detail in Amir et al. (2014) and Aage et al. (2014). In TopOpt 3D, we use the solver without the premature termination heuristics presented in Amir et al. (2014), and instead use a default of three multigrid levels and a relative tolerance of 10^{-5} as convergence criteria for the linear solver.

The possibility of interacting with an ongoing optimization process poses several additional requirements to the optimization kernel. This means a number of safety checks needs to be conducted between each design cycle to ensure that the design problem is feasible, that there are sufficient supports for a given loading, that the FEM system is non-singular, etc. In the rare case when such an issue is detected, the app will pause the optimization and display an error message to the user. Once the problem has been resolved, the app will continue the optimization process.

The optimization kernel runs in its own thread, which makes the GUI respond instantly to user interactions. After each optimization iteration, the optimized shape is sent to the GUI for visualization and the optimization kernel is updated with the user actions, if any.

The achieved performance is 9.9 seconds per optimization iteration when running on a configuration with the highest number of elements ($32\times32\times16$) running on an iPad Air. On a 2.3 GHz Intel Core i7 computer the performance is 1.2 seconds per iteration. The application has the same grid size options on all platforms in order to be able to share models across platforms.

The optimization kernel is currently single threaded, so a future extension is to utilize all hardware threads on the platform to boost performance. At the time of writing, most desktop and laptop computers have two to eight hardware threads, whereas handheld devices usually have two and in rare cases four hardware threads. A related optimization is to use the GPU for some of the computations.

3.2 Interactive shape modelling

A wide number of popular techniques exist to model 3D shapes interactively. The user performs a sequence of modelling operations which change the shape. Often, these operations are performed in a continuous way i.e. across a number of frames where the user has control of the single operation. Some of the frequently used techniques include:

- **Surface-based modelling**, which provides direct manipulation of a polygonal mesh. Operations are performed on vertices, edges or polygons.
- **Solid-based modelling**, which uses parametric geometry. The user will often sketch a profile in 2D and then extrude or sweep this into a 3D shape. Shapes can be combined using boolean operations.
- **Sculpting**, which mimics clay modelling. Typical tools are clay strips, brush, inflate, deflate, crease and smooth.

On the other hand, topology optimization is usually performed in a non-interactive manner, where modelling the problem and inspecting the result are two distinct stages separated by a long wait during which the optimized shape is computed. Often, topology optimization is used as either the only shape design tool, in cases where the aesthetics and design do not matter, or as a post-processing tool, where an existing model can be benchmarked against an optimized shape.

The TopOpt 3D App combines the interactiveness of traditional 3D shape modelling with optimized shape design. To our knowledge, the TopOpt 3D App is the first application that uses this approach for shape design in 3D. Consequently, one of the challenges is to design a simple and user-friendly interface for the application.

[3] On iOS and OSX we use the optimized BLAS and LAPACK bundled in the operating system and on Windows we use AMD's core math library (AMD core math library 2014).

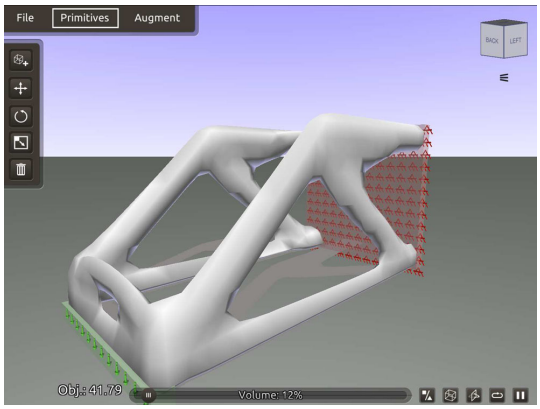


Fig. 2 The user interface with the ViewCube in the upper right corner which shows the current orientation

3.3 User interface design principles

The user interface of the TopOpt 3D App is designed for tablets, smartphones and computers using touchscreen or mouse as input. This imposes some design restrictions, such as different screen sizes, no tooltip help when the mouse cursor hovers over an element and the keyboard should only be used for text input (Fig. 2).

During creation of an appropriate and intuitive cross-platform user interface we have found the following three principles invaluable: simplicity, continuity and consistency. These principles are a subset of the principles of good interaction design described in Tognazzini (2014).

Simplicity principle: Simplicity is often praised as the main principle for good interface design. This is in particular true for apps running on smartphones with small screen sizes and inaccurate touch input. Consequently, to make the user interface work on all platforms, the application uses icon buttons as the primary user interface element, which only takes up little screen space.

The application uses a single viewport for the 3D content. This is in contrast to the majority of 3D programs, which use multiple viewports. The single view is used for displaying both the topology optimization problem (such as loads and supports) and the optimized shape. This makes it easy to grasp how small changes can affect the optimized shape.

Continuity principle: Instead of using instant state changes, we use short continuous transitions between states. Displaying this transition improves the visual perception of the state change.

An example of this is found when modelling the optimization problem. This is done by specifying regions, which are later augmented to have a specific meaning (See Fig. 3). The regions are defined by inserting 3D models into the design domain. The user can choose from predefined shapes (cube, sphere and plane) and other closed surfaces imported from geometry files. The action of inserting the 3D model is performed as an animation in which the model falls into the scene from the sky. This makes it easy to understand the spatial location of the new region - even when it is partial or fully occluded.

Another example of continuity is when changing a constraint, such as the volume fraction, which gradually causes the shape to evolve to a new optimum over time.

Consistency principle: By designing the user interaction in a consistent way, the user only needs to learn a few concepts that can be applied in multiple contexts.

One example is transformation of regions. First, regions are selected using either a clicking or lasso-selecting gesture. Then, a 3D widget appears in the selection which can be used for translating, rotating or scaling by dragging one of its handles, as described in Bowman et al. (2004).

After modelling, each region can be assigned a single label as a load, a support, or a passive element (forced to be void or material). Essentially, the labelled region is used as a consistent way of labeling nodes (for supports) and elements (for loads and passive elements) inside the region in the discretized design domain.

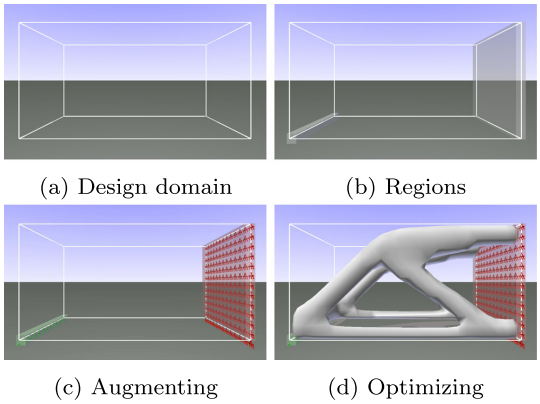


Fig. 3 Workflow: After the design domain has been specified (**a**) regions are specified (**b**) and regions are augmented with loads and supports (**c**) after which the topology optimization begins and the result appears (**b**)

3.4 Visualization and navigation

The result of the optimization kernel is a voxel grid with a density for each element. We use two different methods for visualizing the result; voxel visualization and a marching cubes visualization as seen in Fig. 4. As opposed to the 2D TopOpt App, the TopOpt 3D App does not use the multi resolution approach (Nguyen et al. 2010), since smooth 3D visualization is more practically performed using for

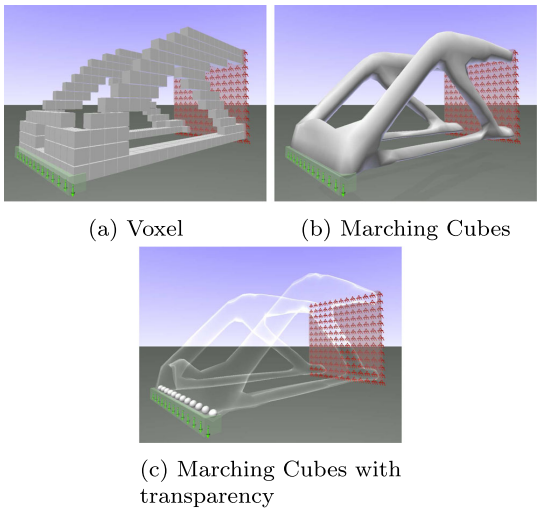


Fig. 4 Visualization modes

example marching cubes. Note that the shape generated by either approach is no longer guaranteed to respect the given volume constraint.

Voxel visualization displays all elements with a density above a given threshold. Voxel visualization gives a blocky-looking result and it is difficult for the user to perceive the shape due to the uniform shading of the planar faces.

A more visually pleasing solution is to use the marching cubes algorithm (Lorensen and Cline 1987) to extract a triangular mesh. The extracted mesh is further improved by removing thin needles, minimizing the curvature energy using an edge-flipping greedy algorithm (Dyn et al. 2001),

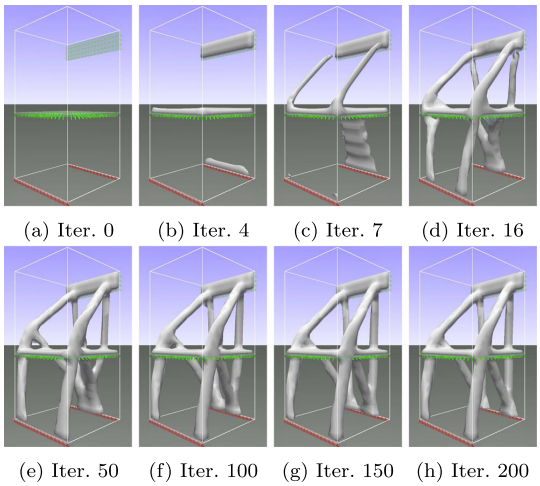


Fig. 5 Modeling a chair (volume fraction 12 %)

Fig. 6 A photo of the topology optimized chair manufactured using a 3D printer

and finally computing the angle-weighted vertex normals. The mesh is rendered using either smooth shading (Fig. 4b) or additive blending where the contours of the surface are highlighted (Fig. 4c).

The opaque visualizations are rendered with two oppositely directed light sources; a sun light and a darker directional ambient light. Furthermore, the scene has a simple skybox where the ground and the sky are used as a visual cue of the orientation of the camera. To help the user perceive the shape, a shadow is rendered on the ground.

The ViewCube (Khan et al. 2008) is used both for navigation and to show the current camera orientation (see Fig. 2). The navigation is performed by rotating the virtual camera around the design while looking at the center of the

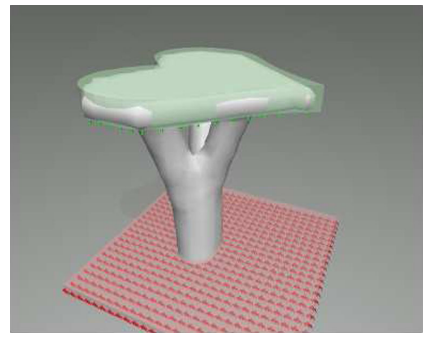
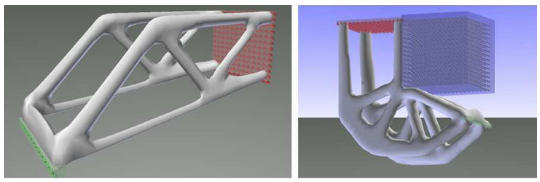


Fig. 7 Shape based on imported heart-shaped geometry



(a) Cantilever (b) L-shaped cantilever

Fig. 8 Classic topology optimization examples

design domain. The ViewCube supports both dragging gesture, which changes the elevation and azimuth angle, and clicking on one of the faces, edges or corners of the cube, which rotates to the selected view.

4 Results

Figure 5 shows how a chair can be modelled:

- Two regions of supports on the ground are defined.
- The seat is modelled using a horizontal plane with vertical loads.
- The back-rest is modelled using a vertical plane with the load direction of the back pointing backwards.

The shape of the chair appears after only 16 iterations. After approximately 150 iterations the chair changes topology from a three-legged chair into a four-legged chair.

The chair can be modelled in 2-3 minutes using around 60 user actions (clicks or drags). About three-quarters of the time is spent on actual modelling and the remaining time the user watches the shape evolve.

Note that Fig. 5 does not show the interactive nature of the modelling. In a typical use-case the problem will be modified frequently during the optimization until a desired shape is found. The optimized chair has been exported from the app and printed in 3D (Fig. 6).

Figure 7 shows how imported shapes can be used for modelling problems that would otherwise be cumbersome to create.

Finally, Fig. 8 shows the modelling and shape of two standard examples in topology optimization; a cantilever beam and an L-shaped cantilever.

5 Conclusion

In this paper, we have presented TopOpt 3D, the first topology optimization application of its kind which runs on both computers and handheld devices. The application shows a new way of modeling shapes where the user does not design the shape. Rather, the user specifies the constraints the shape must obey, and the shape emerges through topology optimization. Since the result of the optimization is displayed interactively, if the result is not as desired, the constraints may be changed during the optimization process, allowing for fast design iterations.

TopOpt 3D is mainly created as an educational tool for teaching topology optimization using a hands-on approach. By using the app students will learn the concepts of topology optimization in a fun and intuitive way. The application can also be used as a tool for prototyping interesting designs early in the design process in e.g. architecture and industrial design engineering.

There are several features which we plan to add to the App such as multiple load cases and gravitational loads which both exist in the TopOpt App (2D). In the longer term, we note that direct manipulation of the geometry is not possible using the presented work. We do, however, believe it to be possible to combine direct manipulation (sculpting) with optimization based shape modeling as discussed in this paper, and plan to pursue that idea.

Acknowledgments The authors gratefully acknowledge the support from the Villum foundation through the NextTop project. The authors would also like to extend their gratitude to the members of the DTU-TopOpt group for their invaluable input on the design and testing of the TopOpt 3D App.

References

AMD core math library (2014). http://developer.amd.com/tools-and-sdks/cpu-development/cpu-libraries/amd-core-math-library-acml/

BLAS Basic Linear Algebra Subprograms (2014). http://www.netlib.org/blas/

LAPACK - Linear Algebra PACKage (2014). http://www.netlib.org/lapack/

Aage N, Andreassen E, Lazarov B (2014) Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. Structural and Multidisciplinary Optimization, pp 1–8. doi:10.1007/s00158-014-1157-0

Aage N, Nobel-Jørgensen M, Andreasen C, Sigmund O (2013) Interactive topology optimization on hand-held devices. Struct Multidiscip Optim 47(1):1–6

Amir O, Aage N, Lazarov B (2014) On multigrid-CG for efficient topology optimization. Struct Multidiscip Optim 49(5):815–829. doi:10.1007/s00158-013-1015-5

Bendsøe M, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224

Bendsøe M, Sigmund O (2003) Topology optimization: theory, methods and applications. Engineering online library. Springer

Bowman DA, Kruijff E, LaViola Jr JJ, Poupyrev I (2004) 3D User Interfaces: Theory and Practice. Addison-Wesley Professional

Davis T, Duff I, Amestoy P, Gilbert J, Larimore S, Natarajan EP, Chen Y, Hager W, Rajamanickam S (2014) Suite sparse: a suite of sparse matrix packages. http://www.cise.ufl.edu/research/sparse/SuiteSparse/

Dyn N, Hormann K, Kim SJ, Levin D (2001) Optimizing 3d triangulations using discrete curvature analysis. Mathematical methods for curves and surfaces, pp 135–146

Khan A, Mordatch I, Fitzmaurice G, Matejka J, Kurtenbach G (2008) ViewCube: a 3d orientation indicator and controller. In: Proceedings of the 2008 symposium on Interactive 3D graphics and games. ACM, pp 17–25

Lorensen WE, Cline HE (1987) Marching cubes: A high resolution 3d surface construction algorithm. In: ACM Siggraph Computer Graphics, vol 21. ACM, pp 163–169

Miller RB (1968) Response time in man-computer conversational transactions. In: Proceedings of the December 9-11, 1968, fall joint computer conference, part I. ACM, pp 267–277

Nguyen TH, Paulino GH, Song J, Le CH (2010) A computational paradigm for multiresolution topology optimization (mtop). Struct Multidisc Optim 41:525–539. doi:10.1007/s00158-009-0443-8

Nielsen J (1993) Usability engineering. Academic Press Inc

Nielsen J (1994) Usability engineering. Elsevier

Rooney B (2011) The wall street journal: ipad 2 more powerful than 1990s supercomputer. http://blogs.wsj.com/tech-europe/2011/05/11/ipad-2-more-powerful-than-1990s-supercomputer/. Accessed 30 June 2014

Sigmund O (2001) A 99 line topology optimization code written in Matlab. Structural and Multidisciplinary Optimization (1999), pp 120–127

Sigmund O, Maute K (2013) Topology optimization approaches. Struct Multidiscip Optim 48(6):1031–1055

Tcherniak D, Sigmund O (2001) A web-based topology optimization program. Struct Multidiscip Optim 22(3):179–187

Tognazzini B (2014) First principles of interaction design (revised & expanded). http://asktog.com/atc/principles-of-interaction-design/. Accessed 5 Oct 2014

Appendix C

Paper C: Improving topology optimization intuition through games

Nobel-Jørgensen, Morten, Malmgren-Hansen, D., Bærentzen, J. A., Sigmund, O., Aage, N. (2015). Improving topology optimization intuition through games. (Submitted)

Noname manuscript No.
(will be inserted by the editor)

Improving topology optimization intuition through games

Morten Nobel-Jørgensen · David Malmgren-Hansen · J. Andreas Bærentzen · Ole Sigmund · Niels Aage

Received: date / Accepted: date

Abstract This paper describes the educational game, TopOpt Game, which invites the player to solve various optimization challenges. The main purpose of gamifying topology optimization is to create a supplemental educational tool which can be used to introduce concepts of topology optimization to newcomers as well as to train human intuition of topology optimization. The players are challenged to solve the standard minimum compliance problem in 2D by distributing material in a design domain given a number of loads and supports with a material constraint. A statistical analysis of the gameplay data shows that players achieve higher scores the more they play the game. The game is freely available for the iOS platform at Apple's App Store and at `http://www.topopt.dtu.dk/?q=node/909` for Windows and OSX.

Keywords Interactive · Topology optimization · Gamification · Smartphones · Tablets · PDE constrained optimization

1 Introduction

Topology optimization has for the past two and a half decade made a great impact on the design of structures and mechanical elements. The method is used in many different fields of engineering and can be applied on many different scales, from designing microstructures to large-scale constructions such as ships, skyscrapers and aircrafts. Topology optimization is usually performed as a discrete step in the design process - often integrated with CAD software. Even though topology optimization algorithms are able to find good solutions to most problems, it is important for their users to have a good intuition for the method in order to get a feel of the process and be able to identify cases where the algorithms clearly get stuck at a local minimum. In cases where topology optimization is not used in the design process, due to time or resource constraints, the final design relies on performance of the "human topology optimization", where a good intuition is critical.

This article describes how we have transformed the topology optimization problem into a game in order to train human intuition for the problem. The game can be used as an educational game in topology optimization lectures – or it can be played by people with no experience in the field. By anonymously tracking players' progress in the game we are able to estimate people's topology optimization intuition and how this intuition progresses the more times a single user plays.

By gamifying topology optimization we are also aiming at heightening the awareness of the field for a broader audience.

2 Related work

Gamification, where game elements and game design are added to non-game contexts, can be used to improve the user experience and user engagement [4]. Gamifica-

The authors acknowledge the support from the Villum foundation through the NextTop project.

M. Nobel-Jørgensen · D. Malmgren-Hansen · J. A. Bærentzen
Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Asmussens Alle,
B.305, 2800 Kgs. Lyngby, Denmark
E-mail: mono@dtu.dk

N. Aage · O. Sigmund
Department of Mechanical Engineering, Solid Mechanics,
Technical University of Denmark, Nils Koppels Alle,
B.404, 2800 Kgs. Lyngby, Denmark

tion is used in various places, including science, where it has helped research in protein folding. Computing how proteins fold is a hard non-convex optimization problem, and finding good foldings is crucial for understanding, and potentially curing, diseases like Alzheimer's, Parkinson's and some types of cancers. Two famous gamification projects using protein folding are:

- **Folding@home** turns consumers' computers into one big distributed computer by utilizing the computer's idle time to perform the heavy computations. Folding@home uses a scoreboard (both personal and team-based) as a way to motivate people to run the software on their computer [2].
- **FoldIt** has a different take on protein folding. The creators have turned protein folding into a game by abstracting the mathematical problem into easily understandable metaphors. In FoldIt, users compete (individually or in groups) to come up with the best folding. In some cases, the player performance can even beat the solutions found by existing protein algorithms. User behavior in FoldIt has been studied in order to extract strategies to improve the protein folding algorithms [5].

The two protein-folding projects also have the positive side effect of increasing people's awareness of the protein-folding problem and its related diseases.

The work presented here is related to the TopOpt app [1], where topology optimization is solved in an interactive manner. Some elements of the user interface have been reused as well as parts of the topology optimization kernel. One feature that has not been reused is the multiresolution topology optimization scheme MTOP [6], since this method does not penalize solutions which are not watertight. This had the implication that solutions where the material was separated by an empty row or column on the fine grid were evaluated as if they were connected.

The TopOpt Game app was launched for iOS and PC on the 28th of August 2014.

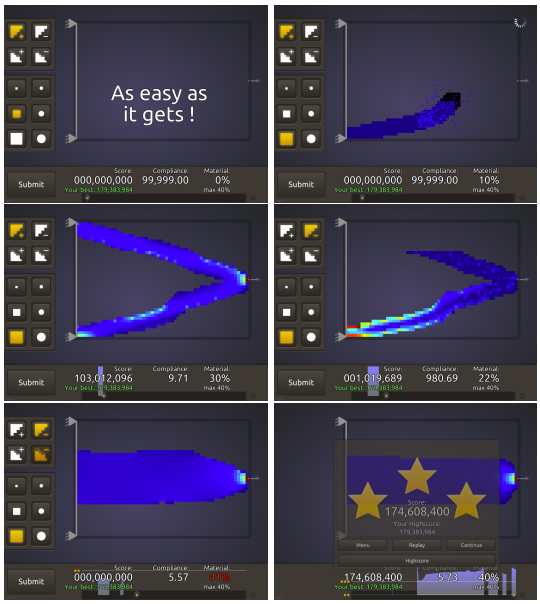


Fig. 1: Gameplay: While the user paints a solution the game provides feedback in terms of a score and by visualizing the strain energy density using the jet color scheme.

$$\begin{aligned} \min_{\rho\in\mathbb{R}^n} \quad & \phi(\boldsymbol{u},\boldsymbol{\rho}) = \boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{F} \\ & V(\boldsymbol{\rho})/V^* - 1 \leq 0 \\ & \rho_i \in \{\rho^{\min}, 1\}\,, i = 1,\ldots,n \end{aligned} \tag{1}$$

where $\phi = \boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{u}$ is compliance, $\boldsymbol{\rho}$ is a vector of n element densities (design variables), $\boldsymbol{u}$ and $\boldsymbol{F}$ are the nodal displacement and force vectors, respectively, $\boldsymbol{K}(\boldsymbol{\rho})$ is the global stiffness matrix, and $V(\boldsymbol{\rho})/V^* - 1 \leq 0$ is the volume constraint. Finally, $\boldsymbol{\rho}^{\min}$ is a lower bound on the design variables.

During level design, a baseline compliance is found using the solver from [1], where each design variable can have a value between ρ^{min} and 1. The baseline compliance is used for rating the player performance in terms of 0-3 stars. The actual rating mapping is adjusted to the difficulty of each problem during level design.

3 Problem formulation

The optimization problem we ask the player to solve in the TopOpt Game is the standard minimum compliance design problem for linear elasticity [3]. Following a finite element (FE) discretization, the classical topology optimization problem can be stated in a discrete form using the density approach [8] as

4 Game design and implementation

TopOpt Game is inspired by puzzle-games (a genre of computer games), which constantly challenge the players and give rewards when progress is made. This en-

Fig. 2: Score graph: Sliding the handle left or right allows easy backtracking.

gagement loop will take the player on a journey starting with simple problems with few supports and a single load and gradually increase the difficulty by adding more loads, restrictions on the design domain, distributed loads and multiple load cases.

The goal of the game is to distribute a constrained amount of material in a design domain in order to minimize compliance (and hence maximize stiffness). The player must find the best material distribution which connects all the loads with the relevant support regions before a timer runs out. Figure 1 shows a typical gameplay. If too much material has been used, the player is penalized by setting the score to zero.

The way material is distributed is inspired by brush strokes in painting programs; The player selects the add or remove material state and a brush type, and then makes a drag-gesture in the design domain. There also exist two specialized tools which add or remove elements only on the boundary of the structure. As a visual cue, a particle effect helps illustrate how elements are constructed or dissolved when using the draw tools. Note that the design elements within the design domain either have material or are void; "graylevel" elements are not allowed in order to simplify the user interaction.

To ensure a good responsiveness of the user interface, we use multiple threads. The main thread is responsible for updating the user interface, listening for events, and rendering the game. Another thread evaluates the compliance of the current structure in an asynchronously way. When the compliance has been evaluated, the value is displayed to the player as well as a score (a scaled multiplicative inverse of the compliance). We found that maximizing a value (score) is a much more intuitive goal than minimizing a value.

The score is the most important user interface element and it is very important that the player is able to see if a change has a positive or negative consequence. For this reason the score is visualized in two complimentary ways:

- **A score label** which accurately shows the exact score to the player as a number. It is easy to compare this with a previous highscore (also shown as a label).
- **A score graph** which allows the player to see the development of the score over time, as shown in 2. Changes to the graph are smoothly animated, which makes it easy to grasp when the score increases. The score graph also shows three important score milestones as horizontal lines for a two-star rating, a three-star rating and previous highscore (if any).

Besides visualizing the score, the score graph also serves two other purposes: It works as the game timer, showing the player how much time remains, and the handle below the graph allows the player to easily backtrack the solution to any previous evaluated state by dragging the handle to a previous time point.

When the player is out of time, or is submitting the level, the score is compared to a baseline score and the player's solution is rated from 0 to 3 stars. We use this simplified rating system to give a clear indication of the player performance.

The player is also able to compete against himself or herself, trying to beat his/hers best score. In addition, the game is on iOS integrated with GameCenter. Using GameCenter, a detailed leaderboard for each level is available as well as the option of challenging friends if a good solution to a level is found.

When a level has been played, gameplay information is sent to a server. The gameplay information contains all information about the player performance during gameplay, including every evaluated action that the player performs as well as score and compliance of each action.

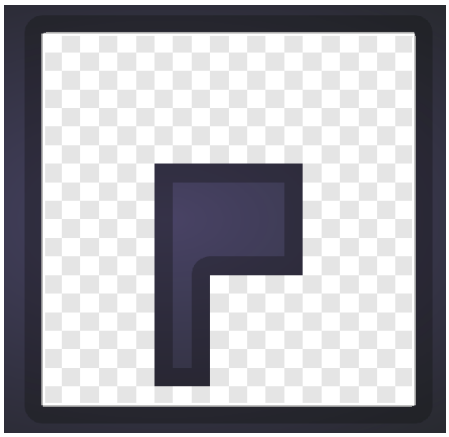
5 Level editor

One of the biggest challenges in creating the game was how to create interesting and fun topology optimization problems. One option would be to create procedurally generated levels containing topology optimization problems created by parameters such as difficulty and time complexity. However, we found that this idea would be a too hard problem to solve and instead decided to manually craft the levels.

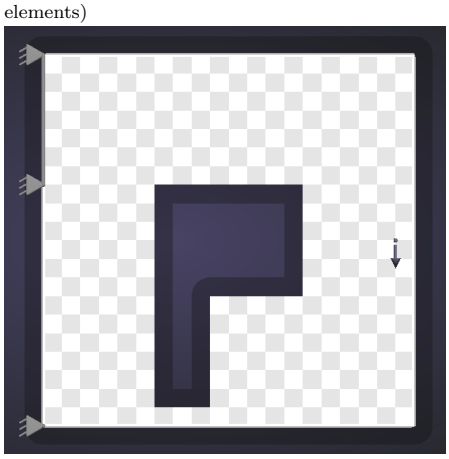
To make level design easy, we have built a level editor similar to the user-interface used in TopOpt app [1].

A user can create a level by defining the size of the design domain followed by removing or adding elements using a drawing tool.

The level is then augmented with loads and supports, which are inserted onto the nodes of the grid and optionally distributed horizontally or vertically. Figure 3 shows the steps of designing a level.



(a) Defining the design domain (size and excluded elements)



(b) Adding loads and supports

Fig. 3: Designing a level in the built-in level editor

When a level has been designed and some additional properties (time, name, category, volume fraction) have been specified it can be play-tested to find out if it works as intended.

User-generated levels are private and only visible to the player who created them. However, the user has the option to suggest a custom level as a global level, and in time this will increase the number of global levels.

Members of the DTU TopOpt group created the initial global levels in the game. The levels include classic topology optimization problems such as cantilever and L-shaped cantilever. A recent addition to the set of levels is the Zhou-Rozvany problem [9], which is interesting since it is one of the few cases where we know the global optimum and where many numerical TopOpt approaches fail.

6 Results

To evaluate whether playing the game actually improves players' intuition about topology optimization, we have analyzed the gameplay data to find a relationship between the number of games a person has played and the score he achieves. Since the analysis is performed on data from uncontrolled usage, we make the following assumptions:

1. Each player (registered or unregistered) corresponds to a single person.
2. In each completed gameplay the player strives for the maximum score.
3. There is a correlation between obtaining a high score in the game and having a good intuition about topology optimization. Hereby we neglect the performance gain of both learning the user interface, as well as known solutions from previous gameplays of the same or similar kind.

The following analysis is based on gameplay data from the global levels. It consists of gameplay observations with the variables listed below:

- Player ID
- Level ID
- Experience (number of games the player has played prior to current gameplay).
- Score (reciprocal of compliance).
- Normalized score (The score divided by the maximum score achieved at the current Level ID).

The normalized score was introduced to remove the effect of different complexities across levels.

A gameplay sometimes has the final score of 0. This usually means that the player broke the volume constraint at end of the game and did not have time to undo the action. We found that these cases do not tell us much about the player's topology optimization intuition and therefore removed such observations from the analysis.

In fig 4 the normalized scores of all observations are plotted as a function of experience. It is seen how the density of the observations decreases with experience. As players individually decide how many gameplays they play, our observations are not balanced at all experiences. In fact only 13 players have played more than 50 times at the time of writing. This gives an increased uncertainty for higher experiences.
In figure 4 we have applied a Linear Mixed-Effect (LME)[1]

[1] Fitted with R-library lmerTest

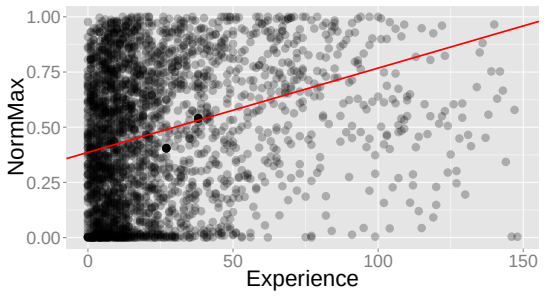


Fig. 4: Normalized score based on experience with Linear Mixed-Effect model fit (red line).
$NormScore = 0.0038 * Experience + 0.3861$

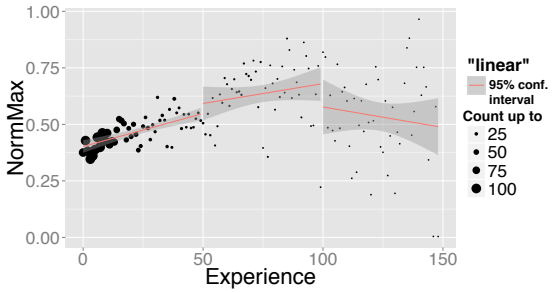


Fig. 5: Linear Regression on averaged data with 95% confidence intervals. LM fits in intervals of 50 experience. Linear Model from 1 to 50:
$slope = 0.003 (p_{value} = 0.01)$
$intercept = 0.405 (p_{value} < 0.001)$
$R^2 = 0.44$

model to the observations. The players cannot be assumed to have equal skills prior to playing this game, as some are already familiar with topology optimization while others are new to the topic. This might affect their starting score, but also their learning curve (increase of score as a function of experience). An LME model is suitable to adjust for this, as it can take into account that a variable has independent variance.

Even though we achieve significant parameters (<0.001) in the model for both slope and intercept, it is visually clear that the data is somewhat noisy. This is due to the fact that, even though players averagely improve with experience, they tend to still have occasionally "bad" gameplays with low score (see figure 4). Because of this, we analyze the data with a different approach as well. In figure 5 we have averaged all players for each experience level and instead of the LME approach we fit a standard Linear Model. Since the density of the original observations is so poor for experiences over 50 (seen by the size of dots in figure 5), we have shown a linear regression separately for experience intervals of 50. From the plot of confidence intervals on the figure, it is clearly seen how much uncertainty there is to modelling the higher experiences.

Table 1 shows the normalized score averaged across all players in experience intervals of 10 steps. The variance within the interval is also calculated along with the count of observations.

Experience	mean	var	obvs
10	0.4016	0.0858	1000
20	0.4611	0.0979	637
30	0.4605	0.0929	347
40	0.5237	0.0943	214
50	0.5202	0.1072	148
60	0.5317	0.0977	84
70	0.6753	0.0693	82
80	0.6515	0.0751	55
90	0.6983	0.0676	36
100	0.6107	0.0879	32
110	0.6101	0.0656	31
120	0.4938	0.0784	19
130	0.4830	0.0470	16
140	0.5819	0.0504	9
150	0.4927	0.1209	8

Table 1: Normalized score averaged across all players

Table 1 shows the same trend as the linear regression model from Figure 5, with an increasing mean score up to an experience of around 90 gameplays. After 90 gameplays the mean score drops and so do the number of observations.

7 Conclusion

In this paper, we have presented the TopOpt Game, an educational game which allows players to learn topology optimization by finding good solutions to given problems. The game shows a new way of teaching topology optimization, by which the students get familiar with the overall concepts. Using the game as a supplement to traditional MATLAB-based teaching allows students to compete against each other and to get a feel for how hard a problem topology optimization is.

We have shown that players averagely increase their score as they become more experienced. We encountered difficulties analyzing high experience due to lack

of observations here. The trend of increasing score with increasing experience seems very strong up to values of 90. Whether the trend decreases because of a saturated learning curve or actually continues, is not possible to answer with the data collected, but the overall trend is clear.
Other than lack of observations our problems in the analysis might be found in the assumptions we state. Referring to the Results section, assumptions 1 and 2 are critical, but probably not always met. We have no guarantee that a player corresponds to a single person and this leads to a source of uncertainty in our analysis. Also, that players may not always perform, or strive, their best in every single gameplay misled the analysis. Players might be distracted during playing or try a silly solution out of curiosity.

One future simplification of the game is to let the game enforce the volume constraint. This should make the gameplay slightly easier, since players have one less thing to think about and it should have a positive effect on the learning rate when the data is analyzed.

When more gameplay data has been gathered, it could be interesting to investigate the data more thoroughly to unveil the underlying strategies humans use to solve topology optimization. Potentially, this could lead to improvements of existing topology optimization algorithms. It would also be interesting to analyze the data to see if some types of problems are particular hard to solve for humans in order to identify typical pitfalls to be aware of. This includes a further investigation of the Zhou-Rozvany problem where the global optimum is known.
In relation to this, a new experiment could involve a test group of 10-20 persons. Letting these players play the game in a controlled environment would improve the chances of the first two assumptions in section 6 to be true. By testing statistically whether the test groups' performance deviates significantly from the other data, it could help reveal whether the assumptions are right. Furthermore the group could get a set of different topology optimization tasks before and after playing the game in order to see whether it improved their skills in this.

The topology optimization game could also be extended to 3D (similar to TopOpt App 3D [7]), where voxels could be added or removed by using a painting gesture on existing material similar to the popular game, Minecraft. Moving to 3D does add some additional challenges, such as visualizing strain energy density inside a volume.

Acknowledgements The authors would like to extend their gratitude to the members of the TopOpt and NextTop groups at DTU for their invaluable input to the design and testing of the TopOpt Game.

References

1. Aage, N., Nobel-Jørgensen, M., Andreasen, C., Sigmund, O.: Interactive topology optimization on hand-held devices. Structural and Multidisciplinary Optimization **47**(1), 1–6 (2013)
2. Beberg, A., Ensign, D., Jayachandran, G., Khaliq, S., Pande, V.: Folding@home: Lessons from eight years of volunteer distributed computing. In: Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, pp. 1–8 (2009). DOI 10.1109/IPDPS.2009.5160922
3. Bendsøe, M., Sigmund, O.: Topology Optimization: Theory, Methods and Applications. Engineering online library. Springer (2003)
4. Deterding, S., Sicart, M., Nacke, L., O'Hara, K., Dixon, D.: Gamification. using game-design elements in non-gaming contexts. In: CHI '11 Extended Abstracts on Human Factors in Computing Systems, CHI EA '11, pp. 2425–2428. ACM, New York, NY, USA (2011). DOI 10.1145/1979742.1979575. URL http://doi.acm.org/10.1145/1979742.1979575
5. Khatib, F., Cooper, S., Tyka, M.D., Xu, K., Makedon, I., Popović, Z., Baker, D., Players, F.: Algorithm discovery by protein folding game players. Proceedings of the National Academy of Sciences **108**(47), 18,949–18,953 (2011). DOI 10.1073/pnas.1115898108. URL http://www.pnas.org/content/108/47/18949.abstract
6. Nguyen, T., Paulino, G., Song, J., Le, C.: A computational paradigm for multiresolution topology optimization (mtop). Structural and Multidisciplinary Optimization **41**(4), 525–539 (2010). DOI 10.1007/s00158-009-0443-8. URL http://dx.doi.org/10.1007/s00158-009-0443-8
7. Nobel-Jørgensen, M., Aage, N., Nyman Christiansen, A., Igarashi, T., Andreas Bærentzen, J., Sigmund, O.: 3d interactive topology optimization on hand-held devices. Structural and Multidisciplinary Optimization pp. 1–7 (2014). DOI 10.1007/s00158-014-1214-8. URL http://dx.doi.org/10.1007/s00158-014-1214-8
8. Sigmund, O., Maute, K.: Topology optimization approaches. Structural and Multidisciplinary Optimization **48**(6), 1031–1055 (2013)
9. Zhou, M., Rozvany, G.: On the validity of eso type methods in topology optimization. Structural and Multidisciplinary Optimization **21**(1), 80–83 (2001). DOI 10.1007/s001580050170. URL http://dx.doi.org/10.1007/s001580050170

Appendix D

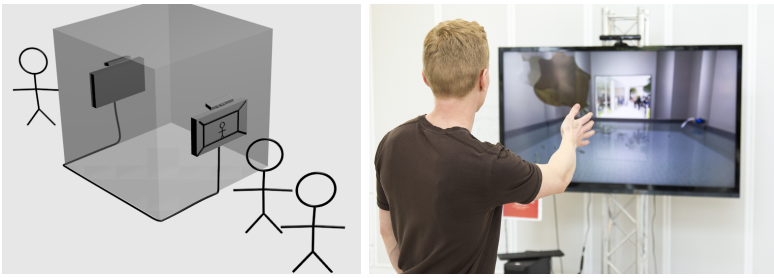
Summary of other publications

The following gives a short description of my additional publications found in the list in the Publications chapter.

Besides the journal article included in Paper A, the TopOpt App was also presented in the talk "Interactive topology optimization" at the 6th European Congress on Computational Methods in Applied Sciences and Engineering in 2012.

I also made contributions to the work of the DSC based Topology Optimization, where Asger Nyman Christiansen was the main author and main contributor. These contributions ranged from discussions of the concepts, implementation details as well as visualization. This work was presented in the WCSMO-10 talk "Topology optimization using an explicit interface representation" and in the journal articles "Topology optimization using an explicit interface representation" (Structural and Multidisciplinary Optimization) and "Combined shape and topology optimization of 3D structures" (Computers & Graphics).

I also gave a poster presentation in Siggraph Asia 2013 with the work of "Pond of Illusion: Interacting through Mixed Reality". The topic presented here was an example of how virtual reality and the real world could be mixed in an



(a) Concepts **(b)** Example usage

Figure D.1: Pond of illusion with multiple views into the virtual space. Notice how it is possible to see out on the other side of the virtual space.

interactive installation which our group had made as a showcase. Figure D.1 shows how the virtual space is created using two screens each with a Microsoft Kinect camera attached.

Finally, I have helped with the paper "Using 3D Models to Annotate SAR Images for Objective Segmentation Performance Measures" where I contributed with creating software for projecting 3D models used for estimating SAR image targets.

Appendix E

Download statistics and user reviews

Table E.1 shows the number of downloads and unique users for the four applications. Web is the Unity Web Plugin, whereas PC is Windows or OS/X applications.

	iOS (*)	Android (*)	Web (#)	PC (#)	Total
TopOpt App (2012)	8,500	4,400	8,200		21,100
FFEM App (2013)	2,100	1,300			3,400
TopOpt 3D App (2014)	2,000			400	2,400
TopOpt Game (2014)	500			100	600
Total	13,100	5,700	8,200	500	27,500

Table E.1: Download (*) and unique user (#) statistics for the applications as of June 2015. Rounded to nearest hundred.

User reviews

The following is a complete list of user reviews from Apples AppStore for iOS and Googles Play app-store for Android apps.

iOS - TopOpt App

AppStore	Stars	Title and Description	Username
Denmark	5	Unique app! Everyone interested in solid mechanics should check out this app!	Sørmand
Denmark	5	Great app. Super educational app - a lot of fun.	C. Niordson
Australia	1	Sub standard. Very poor graphics, substandard app, total waste of time.	Churchiesj
Brazil	5	Amazing App. I'm a Mechanical Engineering student here in Brazil, and this app is very interesting for Mechanics of Solids.	Rafael.Jost
Germany	5	Sehr schön! Macht die Topologie-Optimierung anschaulich und greifbar! (Translated: Very nice! The power topology optimization vividly and tangibly!)	Deejot
Luxembourg	3	Seems to be useful but ... How to change default boundary conditions ???	cedric2080
Netherlands	4	Optimization at your fingertips. Cool little tool to play with. Not yet likely to be used in designing anything, but it sure is fun to see how the shapes come about! Recent updates have made it more versatile and realistic.	AL-hunter
Taiwan	5	Cool. Nice SIMP implementation.	hueyke
Turkey	5	Basarılı. Topolojik optimizasyon konusunda basit ama güzel bir uygulama. (Translated: Successful. Topological optimization application simple but nice.)	ObjektifGorus
USA	5	Fantastic! A very first !	Ben-msu
USA	5	I love it!!! It's amazing to freely reset the boundaries and loading with intuitive click. I love it!!!	figureedge

Average rating: 4.36 / 5.00.

Android - TopOpt App

Stars	Title and Description	Username
5	Us Not the Show original review	pitsanu parabab
5	thanks seeing such an advanced appplication like this is heart warming . This helps both researchers and engineering students to get familiar with an advanced structural design philosophy . i hope to see a more video lectures about this applucation and a more clarifying tutorial.	Mohamed Abdellah
5	Optimal	Kai Habermehl
5	Best! It's so amazing!!	Sonbyeongjin
5	Dr.optim	Ifjohn
5	I am sure that you will like it.	Jiefan Zhao
5	Amazing	Kishen Chatra
5	Nice	Abdullah Waseem
5	Excelente (Translated: Excellent)	Andrés Quiceno
5	Super optimisation tool Very good tool to give awareness for optimisation capabilities. Works excellent.	Matthias Hegenbart
5	Cool!	Mariusz Kuzniar
5	Absolutely amazing. Great code and it runs in a smartphone.	Daniel Häffelin
5	Can't exit. I had to "hard reboot" to exit this app because it fails to follow android standards. It was otherwise cool.	Rebecca Brannon
5	Great toy. Nice tool - great for teaching the basics of topology optimization to students	Niels Aage
5	Excellent app. Have used the website for years and have been even more impressed by this implementation. The only way I could see it being better would be if you could have an image in the background to help you setup the geometry.	Richard Gowland
5	Good	A Google User
5	Epic	A Google User
5	Amazing.	A Google User
5	Great on my Asus Transformer Prime. Works really well on my Asus Transformer Prime tablet.	A Google User
5	Awesome! Great to show off in fron of fellow optimization researchers. Real-time operation is definitely the strongest point of this app. Heartily recommend.	A Google User
5	Optimal app. The app can be used to optimize the design of an unlimited number of structures. Only your imagination limits you! And it only takes a short time to learn how to use it, which is nice.	A Google User
5	Cool. How cool that I can do structural optimization on my smart phone	A Google User

Average rating (based on 75 reviews): 4.65 / 5.00

iOS - FFEM

AppStore	Stars	Title and Description	Username
USA	5	Cool. Very Nice , useful. Thanx	2033 gf
Italy	4	Very funny. Very funny app. The feature that allow to fix and lock imposed displacement is very nice. For 5 stars if there be also spring and load or dumper.	Marco Castelli
Germany	5	Good worke. 2. good App Keep going :D	Fisch3g
Denmark	5	FFEM. Very good idea and it makes it easier to undestand what is happening in the specimen. Well done.	Betonmanden
Denmark	5	Cool! Great tool for illustrating and understanding basic mechanics!	Sørmand

Average rating: 4.8 / 5.00.

Android - FFEM

Stars	Title and Description	Username
5	Cool! Great for first year mechanical / civil engineering students and others interested is solid mechanics!	Søren Madsen
5	A great little app It is great fun to manipulate the small, simple little structures that you can make in this app simply by the touch of your fingers and see the results right away.	Mads Rune Lykke Christensen
5	Great tool! Great tool for illustrating and understanding basic mechanics.	Nis Peter Lange

Average rating (based on 20 reviews): 4.00 / 5.00

Bibliography

[AAL14] Oded Amir, Niels Aage, and Boyan S Lazarov. On multigrid-cg for efficient topology optimization. *Structural and Multidisciplinary Optimization*, 49(5):815–829, 2014.

[BA05] J Andreas Baerentzen and Henrik Aanaes. Signed distance computation using the angle weighted pseudonormal. *Visualization and Computer Graphics, IEEE Transactions on*, 11(3):243–253, 2005.

[Bat08] Klaus-Jürgen Bathe. Finite element method. *Wiley encyclopedia of computer science and engineering*, 2008.

[Ben89] M. Bendsøe. Optimal shape design as a material distribution problem. *Structural Optimization*, 1:193–202, 1989.

[Ben15] The computer language benchmarks game. `http://benchmarksgame.alioth.debian.org/u32/compare.php?lang=v8&lang2=gcc`, 2015.

[Bor01] Jan O Borchers. A pattern approach to interaction design. *Ai & Society*, 15(4):359–376, 2001.

[Bri15] Peter Bright. The web is getting its bytecode: Webassembly. `http://arstechnica.com/information-technology/2015/06/the-web-is-getting-its-bytecode-webassembly/`, 2015.

[BS99] Martin P Bendsøe and Ole Sigmund. Material interpolation schemes in topology optimization. *Archive of applied mechanics*, 69(9-10):635–654, 1999.

[BS03] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods and applications*. Springer Science & Business Media, 2003.

[C+07] Robert D Cook et al. *Concepts and applications of finite element analysis*. John Wiley & Sons, 2007.

[CAS14] Anders Clausen, Niels Aage, and Ole Sigmund. Topology optimization with flexible void area. *Structural and Multidisciplinary Optimization*, pages 1–17, 2014.

[CAS15] Anders Clausen, Niels Aage, and Ole Sigmund. Topology optimization of coated structures and material interface problems. *Computer Methods in Applied Mechanics and Engineering*, 290:524–541, 2015.

[CBNJ+15] Asger Nyman Christiansen, J Andreas Bærentzen, Morten Nobel-Jørgensen, Niels Aage, and Ole Sigmund. Combined shape and topology optimization of 3d structures. *Computers & Graphics*, 46:25–35, 2015.

[CBS] Asger Nyman Christiansen, Jakob Andreas Bærentzen, and Ole Sigmund. *Combined Shape and Topology Optimization*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Solid MechanicsInstitut for Faststofmekanik.

[CHH98] José C Castillo, H Rex Hartson, and Deborah Hix. Remote usability evaluation: can users report their own critical incidents? In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, pages 253–254. ACM, 1998.

[CM01] Sandra Cairncross and Mike Mannion. Interactive multimedia and learning: Realizing the benefits. *Innovations in Education and Teaching International*, 38(2):156–164, 2001.

[CNJA+14] Asger Nyman Christiansen, Morten Nobel-Jørgensen, Niels Aage, Ole Sigmund, and Jakob Andreas Bærentzen. Topology optimization using an explicit interface representation. *Structural and Multidisciplinary Optimization*, 49(3):387–399, 2014.

[CSB14] Asger Nyman Christiansen, Ryan Schmidt, and Jakob Andreas Bærentzen. Automatic balancing of 3d models. *Computer-Aided Design*, 58(Januar 2015):236–241, 2014.

[CTAS14] Asger Nyman Christiansen, Daniel A Tortorelli, Niels Aage, and Ole Sigmund. Combined shape and topology optimization for minimization of von mises stress. In *4th International Conference on Engineering Optimization*, 2014.

[DFAB03] Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[DGAJ08] Frederic De Gournay, Grégoire Allaire, and François Jouve. Shape and topology optimization of the robust compliance via the level set method. *ESAIM: Control, Optimisation and Calculus of Variations*, 14(01):43–70, 2008.

[DHKL01] Nira Dyn, Kai Hormann, Sun-Jeong Kim, and David Levin. Optimizing 3d triangulations using discrete curvature analysis. *Mathematical methods for curves and surfaces*, pages 135–146, 2001.

[EKS94] Hans A Eschenauer, Vladimir V Kobelev, and A Schumacher. Bubble method for topology and shape optimization of structures. *Structural optimization*, 8(1):42–51, 1994.

[ES84] K Anders Ericsson and Herbert Alexander Simon. *Protocol analysis*. MIT-press, 1984.

[Fie88] David A Field. Laplacian smoothing and delaunay triangulations. *Communications in applied numerical methods*, 4(6):709–712, 1988.

[FNTP03] Raúl A Feijóo, Antonio A Novotny, Edgardo Taroco, and Claudio Padra. The topological derivative for the poisson's problem. *Mathematical Models and Methods in Applied Sciences*, 13(12):1825–1844, 2003.

[GGM00] S Garreau, P Guillaume, and M Masmoudi. The topological asymptotic for pde systems. 2000.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[Gra03] Ian Graham. *A pattern language for web usability*. Addison-Wesley Amsterdam, 2003.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[JFY07] Per A Jonasson, Morten Fjeld, and Aiko Fallas Yamashita. Expert habits vs. ui improvements: re-design of a room booking system. In *Proceedings of the 21st British HCI Group Annual Conference*

on People and Computers: HCI... but not as we know it-Volume 2, pages 51–54. British Computer Society, 2007.

[Kli80] GT Klincsek. Minimal triangulations of polygonal domains. *Annals of Discrete Mathematics*, 9:121–123, 1980.

[KMF+08] Azam Khan, Igor Mordatch, George Fitzmaurice, Justin Matejka, and Gordon Kurtenbach. ViewCube: a 3d orientation indicator and controller. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 17–25. ACM, 2008.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[MAB10] Marek Krzysztof Misztal, François Anton, and Jakob Andreas Bærentzen. *Deformable simplicial complexes*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of AutomationInstitut for Automation, 2010.

[MB90] Rolf Molich and Ballerup. Heuristic Evaluation of User Interfaces. *CHI '90 Proceedings*, (April):249–256, 1990.

[MB12] Marek Krzysztof Misztal and Jakob Andreas Bærentzen. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.*, 31(3):24:1–24:12, June 2012.

[MEB+14] M.K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B.B. Christensen, J. Andreas Bærentzen, and R. Bridson. Multiphase flow of immiscible fluids on unstructured moving meshes. *Visualization and Computer Graphics, IEEE Transactions on*, 20(1):4–16, Jan 2014.

[Mle92] H. P. Mlejnek. Some aspects of the genesis of structures. *Structural Optimization*, 5:64–69, 1992.

[Nie93] Jakob Nielsen. Response times: The 3 important limits. `http://www.nngroup.com/articles/response-times-3-important-limits/`, 1993.

[Nie95] Jakob Nielsen. 10 usability heuristics for user interface design. `http://www.nngroup.com/articles/ten-usability-heuristics/`, 1995.

[Nie13] Jakob Nielsen. Windows 8 — disappointing usability for both novice and power users. `http://www.nngroup.com/articles/windows-8-disappointing-usability/`, 2013.

[Nor02] Donald A Norman. *The design of everyday things*. Basic books, 2002.

[NPSL10] Tam H Nguyen, Glaucio H Paulino, Junho Song, and Chau H Le. A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, 41(4):525–539, 2010.

[OSSJ09] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[PWLSH13] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph.*, 32(4):81:1–81:10, July 2013.

[RFR95] John Rieman, Marita Franzke, and David Redmiles. Usability Evaluation with the Cognitive Walkthrough. pages 387–388, 1995.

[RO00] R Radovitzky and M Ortiz. Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-delaunay triangulation. *Computer Methods in Applied Mechanics and Engineering*, 187(3):543–569, 2000.

[S.15] L. S. The end of moore's law. `http://www.economist.com/blogs/economist-explains/2015/04/economist-explains-17`, 2015.

[Sau13] Jeff Sauro. Rating the severity of usability problems. `http://www.measuringu.com/blog/rating-severity.php`, 2013.

[SB11] Mathias Stolpe and Martin P. Bendsøe. Global optima for the zhou–rozvany problem. *Structural and Multidisciplinary Optimization*, 43:151–164, 2011. 10.1007/s00158-010-0574-y.

[She02] Jonathan Richard Shewchuk. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. *Unpublished manuscript*, 65, 2002.

[Shn92] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*, volume 2. Addison-Wesley Reading, MA, 1992.

[Sig97] Ole Sigmund. On the design of compliant mechanisms using topology optimization. *Journal of Structural Mechanics*, 25(4):493–524, 1997.

[Sig07] Ole Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424, 2007.

[SM13] Ole Sigmund and Kurt Maute. Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, 2013.

[SS01] Mathias Stolpe and Krister Svanberg. An alternative interpolation scheme for minimum compliance topology optimization. *Structural and Multidisciplinary Optimization*, 22(2):116–124, 2001.

[SS03] M. Stolpe and K. Svanberg. Modelling topology optimization problems as linear mixed 0–1 programs. *International Journal for Numerical Methods in Engineering*, 57(5):723–739, 2003.

[Sva87] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. *International journal for numerical methods in engineering*, 24(2):359–373, 1987.

[SZ99] J Sokolowski and A Zochowski. On the topological derivative in shape optimization. *SIAM Journal on Control and Optimization*, 37(4):1251–1272., 1999.

[Tid10] Jenifer Tidwell. *Designing interfaces*. O'Reilly Media, Inc., 2010.

[Tog14] Bruce Tognazzini. First principles of interaction design. `http://asktog.com/atc/principles-of-interaction-design/`, 2014.

[TS01] D Tcherniak and Ole Sigmund. A web-based topology optimization program. *Structural and multidisciplinary optimization*, 22(3):179–187, 2001.

[UIM12] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86:1–86:11, July 2012.

[WLS11] Fengwen Wang, Boyan Stefanov Lazarov, and Ole Sigmund. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6):767–784, 2011.

[YC94] RJ Yang and CH Chuang. Optimal topology design using linear programming. *Computers & structures*, 52(2):265–275, 1994.

[ZR91] M. Zhou and G. I. N. Rozvany. The COC algorithm, part II: Topological, geometry and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1-3):309–336, 1991.